

AMERICAN UNIVERSITY OF BEIRUT

AUTOMATED STOCK PRICE PREDICTION USING MACHINE
LEARNING

by
MARIAM IBRAHIM MOUKALED

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Science
to the Department of Computer Science
of the Faculty of Arts and Sciences
at the American University of Beirut

Beirut, Lebanon
October 2019

AMERICAN UNIVERSITY OF BEIRUT

AUTOMATED STOCK PRICE PREDICTION USING MACHINE LEARNING

by
MARIAM IBRAHIM MOUKALLED

Approved by:

Wassim El Hajj, Associate Professor
Computer Science



Advisor

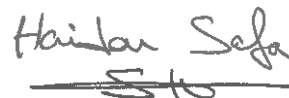
Mohamad I Jaber, Associate Professor
Computer Science

on behalf of Dr. Jaber (email sent)



Member of Committee

Haidar Safa, Professor
Computer Science



Member of Committee

Date of thesis defense: October 17, 2019

AMERICAN UNIVERSITY OF BEIRUT


THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Moukalled Mariam Ibrahim
Last First Middle

Master's Thesis Master's Project Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes
after : **One ---- year from the date of submission of my thesis, dissertation, or project.**
Two ---- years from the date of submission of my thesis, dissertation, or project.
Three ---- years from the date of submission of my thesis, dissertation, or project.



Signature Date

17/12/2019

AN ABSTRACT OF THE THESIS OF

Mariam Ibrahim Moukalled for Master of Science
Major: Computer Science

Title: Automated Stock Price Prediction Using Machine Learning

The financial market is a dynamic and composite system where people can buy and sell currencies, stocks, equities and derivatives over virtual platforms supported by brokers.

The stock market – also known as the equity market – is considered one of the most dynamic components of the free-market economy. It allows investors to own shares in public companies through trading, either by exchange or over-the-counter markets. Thus, giving them the opportunity to make money by investing small amounts of capital which makes for a low-risk endeavor (initially at least), as opposed to opening a new business. Stock markets are however affected by a multitude of factors, making them uncertain and volatile. Which is why we have invested enormous amounts of time and effort trying to predict stock price movements, for by understanding those complex movements can be highly rewarding.

Humans are capable of taking orders and submitting them to the market, but automated trading systems (ATS) on the other hand – implemented via a computer program – can perform quite better and with higher momentum when submitting orders. In order to evaluate and control the performance of these ATSs however, risk strategies and safety measures must be implemented; strategies that are based on human judgement. Many factors are incorporated and considered when developing an ATS, for instance, trading strategy to be adopted, complex mathematical functions that reflect the state of a specific stock, machine learning algorithms that enable the prediction of the future stock value, specific news related to the stock being traded, and so on.

In order to predict market movements, investors traditionally analyzed stock prices and indicators, in addition to news about these stocks – the latter being of utmost importance when trying to understand stock price movements. Most of the previous work in this industry has focused either on labeling the released market news (positive, negative, neutral) and showing their effect on the stock price or looking at historical price movements and predicting future changes. In this work, we adopt a different approach that integrates both market news and stock prices in a single model, for the purpose of achieving better stock prediction accuracy. We propose an automated trading system that uses mathematical functions, machine learning, and other external factors

such as news' sentiments to issue profitable trades. More specifically, we aim to determine the price or the trend of a certain stock for the upcoming end-of-day by considering the first trading hours of that same day. To achieve this goal, we have trained traditional machine learning algorithms, and implemented multiple deep learning models - taking into consideration the relevance of the news. Various experiments were conducted, with the highest accuracy being (82.91%) - and achieved using SVM for AAPL stock.

CONTENTS

ABSTRACT.....	v
LIST OF ILLUSTRATIONS.....	x
LIST OF TABLES.....	xii
Chapter	
1. INTRODUCTION.....	1
1.1. Stock Market Definition.....	1
1.2. Factors Affecting Markets.....	1
1.3. Thesis Objective.....	4
2. LITERATURE REVIEW AND BACKGROUND.....	5
2.1. Efficient Market Hypothesis.....	5
2.2. Random Walk Theory.....	6
2.3. Machine Learning	7
2.3.1. Supervised Machine Learning.....	7
2.3.2. Unsupervised Machine Learning.....	7
2.3.3. Semi-Supervised Machine Learning.....	8
2.4. High Frequency Trading.....	9
2.5. One Day Ahead Price Movement.....	12
2.6. Price Direction Prediction based on financial news.....	15
2.7. Structured Events to predict daily Stock Price Movement.....	17
2.8. Social Media and Stock Price Prediction.....	20

2.9. Integrating Stock market Information and Market News.....	21
3. PROPOSED METHOD.....	25
3.1. Data Sources	25
3.2. Data Pre-Processing.....	26
3.3 Align News with Tick Data.....	26
3.4. Features Generation.....	27
3.5. Data Normalization.....	29
3.6. Deep Neural Network.....	29
3.7. Recurrent Neural Network.....	32
3.8. Support Vector Machine.....	38
3.9. Support Vector Regression.....	40
4. RESULTS AND DISCUSSION.....	42
4.1. Data Description	42
4.2. RNN Results.....	45
4.3. DNN Results.....	50
4.4. SVR Results.....	52
4.5. SVM Results.....	56
5. PROPOSED STRATEGY.....	60
5.1. Risk Model	60
5.2. Risk Model Results.....	61
6. CONCLUSION AND FUTURE WORK.....	64
6.1. Implementing Our Predictive Model.....	65

6.2. Future Work.....	66
REFERENCES.....	67

ILLUSTRATIONS

Figure		Page
1.	Neuron Structure.....	30
2.	Neural Network Structure.....	31
3.	Deep Neural Network Structure.....	32
4.	Simple RNN Structure.....	33
5.	RNN Cell.....	34
6.	Vanilla RNN Cell.....	34
7.	LSTM Cell.....	35
8.	GRU Cell.....	37
9.	SVM Illustration.....	40
10.	FB Stock Price.....	43
11.	AMZN Stock Price.....	44
12.	Google Stock Price.....	44
13.	AAPL Stock Price.....	44
14.	AAPL RNN Stock Price Prediction.....	48
15.	AMZN RNN Stock Price Prediction.....	49
16.	FB RNN Stock Price Prediction.....	49
17.	GOOGL RNN Stock Price Prediction.....	49

18.	APPL SVR Stock Price Prediction.....	55
19.	AMZN SVR Stock Price Prediction.....	55
20.	GOOGL SVR Stock Price Prediction.....	55
21.	FB SVR Stock Price Prediction.....	56
22.	Proposed Risk Strategy.....	60

TABLES

Table		Page
1.	Data Description.....	42
2.	AAPL RNN Results.....	46
3.	AMZN RNN Results.....	47
4.	FB RNN Results.....	47
5.	GOOGL RNN Results.....	48
6.	RNN Metrics.....	50
7.	DNN Results.....	51
8.	DNN Metrics.....	52
9.	SVR AAPL Results.....	53
10.	SVR AMZN Results.....	53
11.	SVR FB Results.....	54
12.	SVR GOOGL Results.....	54
13.	SVR Metrics.....	56
14.	SVM AAPL Results.....	57
15.	SVM AMZN Results.....	57
16.	SVM GOOGL Results.....	57

17.	SVM FB Results.....	58
18.	SVM Metrics.....	58
19.	Best Model Accuracy for each Stock.....	59
20.	Results of Trades profits with and without Risk Strategy.....	62

CHAPTER 1

INTRODUCTION

1. Stock Market Definition

The financial market is continuously growing, and is considered as an important source of income, especially when various strategies are adopted to minimize the risk of loss. However, the volatility of the market, the complexity of the factors affecting stock prices, and the sudden news that affect the trends of prices, lead to the hesitation of investors in this business.

2. Factors Affecting Markets

The stock market's volatility can be attributed to several external factors such as dynamicity of the market and complexity of dimensionality; these make predicting the trend/price of a stock a difficult and challenging task even with deep learning models [12]. These external factors are grouped into fundamental factors, technical factors and market sentiments, and can be summarized as follows:

- Supply and demand. If traders for example, tend to buy a stock more than sell it, this affects price - probably by rising it - since demand is higher than supply.
- Stock prices can experience unexpected moves with a single news story, hence moving artificially high or low. Consequentially, investors simply cannot predict what happens to a stock on a day-to-day basis. This is

what we call market sentiment factors, which include company news, economics, and world events.

- Global economy. The flow of money and transactions moves according to the economic situation of traders, which is intertwined with the economy of a country.

- Stock historical prices. Each stock has a range within which tick data moves, and these can be observed when looking into chart patterns and behavior of investors.

- Public sentiments and social media. A tweet from a president, an article release, a CEO resignation, and so on can greatly affect the price of a related stock(s).

- Natural disasters. For example, the “haiti earthquake” that killed around 316,000 people affected the S&P index by going down 6.6% after 18 trading days.

- Earnings Per Share (EPS) is a fundamental factor affecting stock prices. Investors tend to purchase stocks with high EPS, since they expect to gain substantial profits. The demand on a particular stock, company management, market sector dominance, and cyclical industry performance, will result in the movement of a stock price.

- Inflation and deflation are technical factors. Inflation means higher buying prices and thus higher interest rates, which results in a decrease in stock price. On the contrary, deflation means lower buying prices, and thus lower profits and interest rates.

All these diverse factors - and other factors as well - affect price movements, making stock prediction quite difficult. Researchers assume that market prediction does not exhibit random behavior [6]. Many publications have been written on the topic, attempting to increase the accuracy of future price predictions. [3] Studied the influence of public information reported by Dow Jones and concluded that a direct relation does exist between released news articles and stock market activities.

News releases related to a company's activity result in traders assuming an effect on price movement. For instance, when positive news is released, traders will tend to buy which results in stock price increase. On the other hand, when negative news is released, traders tend to sell and thus push stock price down. Although there is no doubt as to the effect of news on traders' actions, only few studies use the news factor in predicting price movement.

In addition to the factors affecting stock market, we also ought to take into consideration the types of investors and trades. They can be divided into two main categories; professional investors that use short-term trading based on the knowledge they have - such as institutions, and the other type which pertains to individual investors who rely on long-term trading given the limitation of their knowledge regarding stock movement. Hence, the strategies of this task should take into consideration these types of trading investors in order to avoid risk of large losses.

Techniques used in predicting prices can fall into 2 main categories as well. First, technical analysis that is based on historical prices and indicators, and second, artificial intelligence which uses a large amount of data from diverse sources.

Stock price prediction can be defined as a time series problem when integrating the time factor, which results in using deep learning models for time series such as Artificial Neural Network, Recurrent Neural Network and a hybrid combination of RNN with LSTM cell. Additionally, SVM can be used for regression and classification problems given it uses Kernel Trick which can handle non-linear problems. These reasons lie behind choosing the models tested in our work.

3. Thesis Objective

In this thesis, we tackle the problem of detecting the price or trend of a certain stock or security taking into consideration unexpected, sudden news, and other hidden factors that highly affect stock market prices. We also focus on short-term trades that have higher risks than long-term trades. Most of the existing strategies and algorithms are based on pure mathematical calculations and machine learning models that might lead to unexpected failures when sudden news is released. Take, for example, the case where Samsung released Note 7 and then recalled it due to battery damage, Samsung share prices witnessed a dramatic decrease. Such an event could not have been captured by strategies relying solely on pure mathematical algorithms.

CHAPTER 2

LITERATURE REVIEW AND BACKGROUND

1. Efficient Market Hypothesis

In early research related to stock market prediction, [5] proposed the Efficient Market Hypothesis (EMH) and [4] proposed the Random Walk theory. These theories proposed that market prices are affected by information other than historical prices, and thus market price cannot be predicted.

The EMH suggests that the price of a stock depends completely on market information and thus any newly released information will lead to a change in price. This theory also claims that stocks are always traded based on their fair value, i.e. traders cannot buy or sell stocks at a special price - undervalued or inflated - and therefore, the only way a trader can increase their profit is by increasing their risk. EMH discusses three different variations that affect market price:

- Weak Form: suggests that prices represent all the information available related to the market, and also considers that the historical prices are independent of future prices. This infers that technical analysis based on old prices will not result in better profit gain.

- Semi-Strong Form: expands the Weak Form to consider the new released information related to the market in addition to prices, therefore this form assumes that prices change rapidly when news are released, and thus fundamental analysis cannot be used to achieve prediction of future price movements due to the release of instantaneous public information.

- Strong Form: states that price movements reflect all public, private, historical and current information available, in addition to prices. So, this form proposes that even non-public information - such as information known to the CEO of a company alone - will surely affect price movement, and such movement cannot be predicted.

EMH states that any price movement is either a result of newly released information or a random move that would prevent prediction models from succeeding.

2. Random Walk Theory

The Random Walk Hypothesis [4] states that stock prices are randomly changed and argues that past price movements are independent of current movements. This is slightly different from EMH as it focuses on short-term patterns of the stock market.

Based on the above two hypotheses by Horne, J. C. et al. (1967) and Fama, E. F. (1970), the stock market will follow a random movement, and the accuracy of predicting such movement cannot exceed 50%.

On the other hand, many recent studies have shown that stock market price movements can be predicted to some degree. These studies depend on two different types of financial analysis in order to predict stock market prices:

- Fundamental Analysis: is based on the health of a company – including qualitative and quantitative factors such as interest rates, return on assets, revenues, expenses and price to earnings among others. The aim of this analysis is to check the long-term sustainability and strength of a company for the purpose of long-term investment.

- **Technical analysis:** is based on time series data. Traders analyze historical price movements and chart patterns and consider time as a crucial parameter in prediction. Technical analysis can rely on three main key factors: stock price movement – though it may often seem random, historical trends which are assumed to repeat as time passes, and all relevant information about a stock.

3. Machine Learning

In most recent studies, different machine learning techniques have been used to predict stock prices. Machine learning (ML) has proven to be a good tool used in price prediction; it uses data analysis techniques to draw a generalized pattern.

Machine learning can be categorized into four main classification types based on the data given:

1. Supervised Machine Learning

Supervised Machine Learning is a method in machine learning that uses labeled sets of data, of x inputs and y outputs to predict the future. The model learns from a given data set and approximates the mapping so that prediction becomes more accurate; this also can define several problems described as classification if we have multiple output classes, and regression if the data is continuous.

2. Unsupervised Machine Learning

Unsupervised Machine Learning unlike supervised learning - uses datasets with inputs only, without outputs, thus the information is neither classified nor labeled. This kind of data used can be named 'non-structured data'. Unsupervised learning has no teacher to learn from - like labeled sets - and can be grouped into clustering or associations. Clustering is about grouping similar data together, and association is about relating data sets to each other.

3. Semi-Supervised Machine Learning

Semi-Supervised Machine Learning is a combination of both types of learning - supervised and unsupervised - where you have a large input dataset and only few are labeled. Many machine learning problems fall under this category, and mixed techniques are used to solve these problems to improve accuracy.

Artificial Intelligence and machine learning have recently been widely used to automate and solve some real-life tasks, given the main component - data - is available now. ML models and AI are based on the availability of data; the more data we have, the better our model can perform. Nowadays, a lot of data can be generated every day from social media, sensors, company's data and bank transactions. Any action we take in our life is being stored for future use by ML and AI. This data can be analyzed to meet our target. ML and AI are techniques that can be used to analyze and manipulate the data. A sample data is used (historical data) to build a model and predict the future based on the pattern generalized from the historical data.

Machine learning is now used in investing and trading tasks, where it can replace human manual tasks with automated tasks performed by machines. As we mentioned

above, we have two types of analysis used in the stock market, one of which is technical analysis which is based on price analysis and previous patterns. So, ML can learn from this pattern and build a model with good performance. Moreover, ML is a pure programmatic task, which grants ML algorithms advantage over human decisions, since a human will be affected by news releases and might get confused as to the decision to make. Also, ML algorithms are completely based on computations which means that luck plays no role. Furthermore, ML and AI algorithms can capture hidden and complex signals that humans would not be able to analyze and access effectively.

Stock market prediction can be defined as a time series problem in ML. A time series data is a sequential data collected over a period of time; this data usually follows a pattern such as the sales of a supermarket over consecutive months. ML supports different models used for time series problems, such as RNN (Recursive Neural Network) where time series data is used as a historical data sample to predict the future.

Different machine learning models and risk strategies have been applied to stock market prediction tasks, mainly trying to predict the direction of a price within different time frames and using various features that would affect market prices.

4. High Frequency Trading

[3] A high frequency trading strategy has been developed - based on 'deep neural network' - to predict the next one-minute average price of APPL Stock. ANN (Artificial Neural Network) have proven their ability to extract features and learn complex patterns of time series data. An ANN with multiple layers is called a DNN (Deep Neural Network) and helps transform data from low level to high

level features. Four main features have been used as inputs for the DNN model, these can be considered as technical analysis features for the stock market given, they are based on mathematical calculations as described below:

- Log Return: is a finance term that represents the logarithmic difference between the closing price at time t and closing price at time $t-1$.
- Pseudo-Log-Return: is the logarithmic difference between average prices of consecutive minutes.
- Trend Indicator: is a linear model applied on 1-minute tick data to generate a linear equation with slope a . A negative slope implies a decrease in the price while a positive slope implies an increase, and a slope close to zero implies that the price is almost stable.

Tick-by-tick data from the TAQ database for APPL stock in NYSE market was collected for about two months in the year 2008 (from September 2 to November 7). The timeframe was chosen on purpose to include a crash in APPL prices, in order to check how the model would act when dealing with a volatile market affected by external factors such as news. Once data was collected, verifying the correctness of that data was step one; to ensure there were no negative values, nor missing days. Then, tick data was formatted to be summarized as one-minute data with the following three time series features: the average price, standard deviation and trend indicator.

After grouping the data to one-minute feature, the input data was formalized as follows: the time feature was included in the inputs as two parameters - minutes and hours - where hours is a number between 0 and 6, and minutes a number between 0 and 59. A variable window size was used for the other inputs - consider the size of the window is n ;

then the input file also included last n pseudo-log-returns, last n standard deviations and last n trend indicators. The output of the model was next; one-minute pseudo-log-return that allows us to calculate the average price by inverting the formula to exponential form. Then after having the input data file ready, it was given to DNN as one input layer, five hidden layers, and one output layer. The neurons of each layer were constructed based on window size as follows: let's assume I is the number of inputs, then the number of neurons will be respectively I , I , $4I/5$, $3I/5$, $2I/5$, $I/5$ and 1 neuron. *tanh* was the activation function used for all neurons except for the output neuron which uses *linear* activation function. The data was fragmented to training and testing data, taking into consideration the crash, so it was divided as follows: the first 50% have a breach and the second 50% almost no-trending; 7.5% of each 50% was considered as testing. An open source software H2O [4] was used to build the model, which uses ADADELTA as an adaptive learning rate algorithm.

To calculate the performance of the model, two parameters were considered: Mean Squared Error (MSE) and directional accuracy.

The model was trained during 50 epochs with different window sizes. The results show that window size 3 features the best performance of the model, with accuracy 66% and 0.07 MSE.

The result of the above model was tested on risk strategy that compares the closing price to the predicted price. If the closing price was greater than the predicted price, then sell and wait until the current price reaches predicted price; and then buy, otherwise wait until the next minute and repeat.

Same thing applies if the closing price is less than the predicted price; then buy and wait for the current price to reach the predicted value. This risk strategy yields approximately 81% of successful trades when applied on testing data frames selected for the collected data.

As a conclusion, a DNN can learn the complexity and dynamicity of the market with a good precision and performance.

5. One Day Ahead Price Movement

[1] Another hypothesis has been done on stock market prediction, to predict one-day-ahead price movement using disparate sources of data - by combining data from online sources with prices and indicators, we can enhance the prediction of the stock market state. The combination of news and crawling sources integrated with prices has never been used by financial experts before. So, the AI algorithm to be used with these disparate sources was also a task to examine. This study was tested on APPL stock information gathered over 3 years from May 1, 2012 to June 1, 2015, with multiple inputs and different output targets. Target 1 is a comparison of open price of day $i+1$, with closing price of day i ; Target 2 is a comparison of open price of day $i+1$, with open price of day i ; Target 3 is a comparison of closing price of day $i+1$, with closing price of day i ; Target 4 is a comparison of closing price of day $i+1$ with open price of day i ; Target 5 calculates the difference in volume traded between day $i+1$ and day i . All these targets were presented as binary classification, where 0 implies a fall in comparison of targets and 1 implies an increase in compared targets. The approach used to predict stock price movement consists of three phases illustrated below:

- In Phase one, four datasets were gathered from disparate sources. The first dataset includes the public information available at Yahoo finance online for stock prices (open, close, high, low volume). In addition to that, DJIA and NASDAQ daily indices were used, and finally, price to earnings ratio was also added to this dataset. The second dataset includes the number of unique page visits to Wikipedia per visitor per day. The third dataset includes a count of data published on Google related to a company on a specific date. And finally, the fourth dataset includes three technical indicators (Stochastic Oscillator, Larry William, Relative Strength index) that represent the variation of stock price over time. Additional features were generated from four datasets to provide a meaningful parameter for the model such as Wikipedia Momentum, Wikipedia rate of change, Google momentum, Google relative strength index, and moving average of stock prices over 3, 5 and 10 days.

- In Phase two, feature selection was done in order to select the best combination of features that could produce a better performance and accuracy. This phase consists of two steps; step one is the definition of different targets, and step two is recursive feature elimination (RFE) aimed at finding the best subset of features that will eliminate model overfitting and get better accuracies. SVM-RFE was used to rank the most important features and eliminate the less ranked features, with different subsets generated for different targets. For each target twenty features were selected as input for the next step. A common observation could be done; that for any

target, all the datasets were represented by at least one feature. Close, open and high prices for the previous day were included for all targets. Moreover, features were listed in descending order for all targets in the following order: predictors from first set, followed by indicators, then Wikipedia features, and finally Google news features. It could be observed from the feature selection step that for all targets, the features are from the four data sources - which means that information gathered from all sources is useful. In addition to the close, high and open prices that were included in all data sets form a significant predictor.

- In phase three, different AI techniques such as Artificial Neural Network (ANN), Support Vector Machines (SVM) and Decision Trees (DT) were applied to predict stock price movement. Compared to each other - based on 10-fold cross validation - the basic structure of each AI can be illustrated below:

- ANN was widely applied in stock market prediction tasks; sigmoid function was the activation function used.

- DT was also used for this task; the model starts by splitting the dataset to a smaller subset, until a decision reaches the final state - C5 algorithm was used.

- SVM is a popular approach in stock market prediction tasks; Radial basis function was the kernel function used in SVM.

To evaluate the above listed models, different parameters were calculated: accuracy, area under the curve (AUC), F- measure, G-mean, Matthews correlation coefficient (MCC), precision and sensitivity.

After evaluation of the three different models listed above, the output comparing open price of day $i+1$ with open price of day i achieved the best prediction accuracy, with around 85% using SVM model. Then to check the usefulness of the data from disparate data sources, different scenarios were applied; starting with basic information only (market data), then adding more features such as indicators, Wikipedia and Google news. Seven scenarios were formed from different combinations of the above, and as a result Wikipedia data showed that it is a stronger predictor than Google news data. As a result, adding features from the above data sources showed the best performance when tested on the same model. This implies that adding data from different external online sources enhances market prediction.

6. Price Direction Prediction Based on Financial News

[6] Another study has been done on the stock market, trying to predict direction of the price based on financial news. The study was done in 2009 as market prediction was and still is facing difficulties due to the ill-defined parameters. In order to use the financial news articles in the prediction model, news should be represented as numerical values. Several techniques have been known to analyze articles related to certain stocks, to label these articles with sentiments or use them as vectors for the input features. These techniques could be bag of words, noun phrases, named entities and proper nouns. The proper noun technique is a combination of noun phrases and named entities and

outperforms other techniques, based on a comparison study. Grouping similar companies was a new point introduced to the research. Different systems have been built to group companies into sectors, sub sectors, industry groups and industries.

AZFin Text is a system built (6) that predicts price changes, 20 minutes after news releases. The main component of this system is the financial news articles collected from Yahoo finance and represented as noun phrases; using online tool (AzteK). The second main component of this system is the stock price data collected in one-minute timeframes. In addition to these two major components, data from trading experts was collected and represented as metrics of recommendation from different systems. Then the final major task after collecting the data was to build the model.

For the first component, the data was collected and represented as noun phrases, taking a certain frequency for each phrase in an article in order to eliminate useless phrases. Then all the collected noun phrases were represented as vector of binary values, indicating the presence or absence of a phrase in the article. To finalize the input of the model, stock price quotations, during the same minute news was released, were added to the input matrix. Add to that, +20 min price which is the output of the system. Then the data was fed to different models based on the categories generated by the GICS system.

A Support Vector Regression (SVR) model was built to predict the price after 20 minutes of news release. Different metrics were used to evaluate the model: such as measure of closeness that evaluates accuracy of actual price with respect to the predicted price through mean squared error (MSE); directional accuracy that evaluates accuracy of the direction of the price; and the final metric is the simulated trading engine that evaluates the difference between the actual price at the news release time and the

predicted price after 20 minutes of the release. If the difference is more than 1%, then a trade will be taken, either buy or sell based on the predicted direction.

The data collected to build this model was gathered from 26 October 2005 until 28 November 2005, which is a 5 weeks' timeframe with 23 trading days. 2908 news articles were collected for this study - only during market time - from 10:30 to 3:40, leaving 1 hour for market opening to show the effect of news released during closure of the market. Moreover, a new constraint was added to the model - only one article could be used for a time period of 20 minutes. So, if two articles were released during the same 20-minute timeframe, both would be discarded. The model was trained per category, sector, sub sector, industry group, industry and for all stocks. Any category with less than 10 news articles was discarded.

The results show that the best accuracy was established for sector-based category with 71.18% average directional accuracy and 8.5% average return. The best sector accuracy was for financials 76% and the worst was for materials with 63%.

It is evident that released news and published articles affect the market. Most of the existing studies that analyze news rely on shallow features such as bag-of-words, named entities and noun phrases.

7. Structured Events to predict daily Stock Price Movement

A new representation was introduced by [7] which represents news as structured events to predict daily stock price movement. Unlike previous approaches, this representation can show the relationship between events; representing phrases as vectors

or bag of words cannot show the actor, action or the actor which the action is applied to, thus trivial representations cannot show the relationship between event and stock.

To evaluate the performance of this new representation, news articles data was collected from Reuters and Bloomberg, in addition to the daily closing prices of S&P index.

As an initial step, the articles should be represented as events. The event is denoted as a tuple (O_1, P, O_2, T) where O_1 is the object conducted, P is the verb which represent the action, and O_2 is the object on which the action was applied at time T. This representation introduces a new analysis for events, which is the relationship between objects, since previous researches tended to represent news as terms without any relationship. After establishing event representation, a new challenging task - considered as the contribution of this paper - is event extraction that requires predefined events and templates. Open IE is an online tool used to extract events without requiring any templates or human work.

An extracted sentence from news text follows the below procedure:

- Event Phrase Extraction: that has two main constraints - syntactic and lexical. These constraints should be satisfied for the longest sequence of words, where the sentence starts with an extracted predicate verb using dependency parser.
- Argument Extraction: that extracts the remaining features of the tuple O_1 which is the first object to the left of the extracted sentence above, and O_2 which is the nearest object to the right.

The final step before building models, is event generalization that relates the extracted action to a class where the class replaces the action in the tuple, after extracting lemma form of inflected words using Wordnet stemmer.

Two different models were built to test the above representation. A linear SVM model which has news document as input, and +1 or -1 as output indicating increase or decrease in the price for different timeframes respectively. A non-linear Deep neural network (DNN) model is also implemented to learn hidden relationships between events. The DNN consists of two hidden layers using sigmoid activation functions for all layers.

Input features for both linear and nonlinear models were the same: bag-of-words features which use the trivial TFIDF representation after removing stop words, and event features represented by different combinations of the tuple $(O_1, P, O_2, O_1 + P, P + O_2, O_1 + P + O_2)$ where O_1 is the first object to the left of the extracted sentence above, O_2 , the nearest object to the right, and P represents the verb. This feature representation is used to reduce the sparseness of the representation in addition to verb classes.

To evaluate the models, different scenarios were applied. In order to test the performance by calculating different metrics such as accuracy and MCC, the collected data was split into three data sets: Training 80%, Testing development 10%, and Testing 10%.

When comparing results of the models with bag-of-words articles representation, structured events showed a better performance. When comparing the models from another perspective, DNN performed better than SVM due to its ability to learn hidden relationships. Moreover, it was shown from different timeframes used (1 day, 1 week, 1

month), that the shorter the frame, the better the results. Thus, the best model was DNN, with structured event features for daily prediction, with accuracy around 60%.

After achieving the above results, varying the number of hidden layers was the second task. So, the model was tested for one, two and three hidden layers, resulting in the best performance for two layers. In addition to the above scenarios, varying data from collected articles also enhanced the results, such as using only title, or only content, or both; combining title and title provided by Bloomberg/Reuters performed best.

The model was then tested, digging specifically per stock such as Google Inc. and taking three different sets of articles: all news, company news and sector news. As a result, company news outperformed all other scenarios, including the S&P index prediction, probably because company news directly affects the volatility of shares, unlike sector and all other news that may be irrelevant.

8. Social Media and Stock Price Prediction

Another important factor that might affect stock market prediction is the data that is being published on social media such as Twitter and blogs; such news is considered trending and an important source for big data. This factor was studied in [9], with results confirming the effect of tweets moods on the stock market. For this study, Twitter data was collected from 28 February 2008 to 19 December 2008 (9,853,498 tweets). Initially, the tweets were filtered to include only tweets with words “I am”, “I feel”, “makes me”, “http” and “www”. Then grouping tweets per day was the second task, after removing stop words and punctuations. These tweets were given to two different tools, to be labeled with the mood generated. The first tool is OpinionFinder that labels a tweet with a Positive

or Negative mood, however the second tool GPOMS categorizes a tweet into 6 different moods (Calm, Alert, Sure, Vital, Kind, Happy). These result in a seven-mood time series. In addition to tweets data, DIJA closing prices were collected from Yahoo finance. In order to determine the relationship between OpinionFinder and GPOMS, regression models were used based on the following equation: $(Y_{OF} = \alpha + \sum_i^n \beta_i X_i + \epsilon_t)$. This model shows that OF and GPOMS are correlated when it comes to Happy, Sure and Vital moods. To test whether the models can predict the change, two linear models were built, where one uses lagged values for DIJA only, and the other one uses lagged values for both DIJA and moods. From these models, it was observed that Calm mood has the best correlation with DIJA movement within a three-day variation. Since the relationship between stock market and public mood is nonlinear, a five-layers self-organization fuzzy neural network was tested on two sets of 3-days-past of DIJA and combined with moods permutations. The input of the neural network consists of DIJA and mood values for the last n days. Based on the correlation between Calm mood and DIJA prices, n was chosen to be equal to three. Training data was used from 28 February to 28 November and the rest of the data was used for testing. Different data permutations were used to check the correlation between different moods, DIJA price, {DIJA, X1}, {DIJA, X1, X2} and so on. Directional accuracy and mean absolute percentage error (MAPE) were used to evaluate the performance of the model. The results showed that the highest accuracy could be established when using calm mood with DIJA variations - for 3 days - to achieve 86.7% accuracy and 1.83 MAPE.

9. Integrating Stock market Information and Market News

Finally, a study that combines most of the above factors for a shorter time frame was done by integrating market news and stock market information [2]. From different recent studies, two factors have shown their effect on the market, news released and tick intraday data. Since, data has kept on increasing, making it even harder to process manually, which led to two main algorithms being introduced by researchers to cast the problem: Classification and Regression. The Classification approach mainly predicts direction of market prices; classifying the problem into two classes (Positive, Negative) or multiple classes (Extremely Positive, Positive, Neutral, Negative, Extremely Negative). Unlike Regression which performs numerical predictions instead of directional; this approach also does not show signal strength bias, same as classification. So, the Multi kernel Support vector regression approach was used in this study for short-term prediction integrating both factors: news and prices to overcome both information bias and signal strength bias. The study was done on data collected from the Hong Kong stock exchange HKEx for HSI index prices in the year 2001, with 28,885 news articles collected for the same year from Caihua - where both news and prices (bid, ask) have time as the basic feature in data collection. For the prices, data considered high frequency and few other points ought to be handled during preprocessing: the huge amount of data we have, and the variation of interval in tick data as some points might be missed.

Two main steps were performed before aligning news data with prices:

- Processing news articles: features to be extracted from news articles after having articles segmented by Chinese software, since the articles collected were in Chinese language, and performing word filtering such as removing stop words and then weighting the resulting data using TF-

IDF. Therefore, each article was represented as a vector of frequencies, and since the vector formalized has a high dimension, feature selection is applied on the vector by selecting the most 10% occurred words in the articles.

- Processing historical prices: prices should be sorted by time since this is a time-series problem, and missing ticks should be handled by one of the following methods, linear time weighted interpolation, or nearest closing price - used in this study - due to its fast performance.

As a second step and after processing news and prices, both sources should be aligned together. For news articles, two things were taken into consideration; only news within market opening time (10:30 – 12:30 and 14:30 – 16:00) were considered, removing first and last 20 minutes which is the maximum frame used in this study. The second thing to be handled was that if 2 news were released during the selected timeframe, one of them should be discarded. The price movement after news released is then calculated as ratio $(p_i - p_0) / p_n$ where i belongs to a set $\{5,10,15,20,25,30\}$ and the ratio is used as a label for the news. Moreover, 30-minute prices before release of the news is used to generate indicators RDP_i (i belong $\{5,10,15,20,25,30\}$), relative strength index (RSI), raw stochastic value (RSV), William index (R), bias (BIAS), psychological line (PSY).

Three vectors were generated: News vector, Indicators vector and Ratio vector. Normalization of these vectors was applied so that the data ranges between 1 and -1. After having the input featured ready, three models were developed; one based on labeled news articles only, using SVR, the second one took only indicators calculated as input for SVR model, and a third model using the combination of both news and prices features as input to a two-sub kernel MKSVR, where each layer handles one vector input.

Three different training/testing sets were generated 8/4, 9/3, and 10/2. Grid Search in a 5-fold cross validation was used to determine the parameters.

From the results obtained, it showed that MKSVR outperforms other experimented models using RMSE and MAE metrics to validate the performance. Also, the model showed best performance for the 5-minute ratio.

CHAPTER 3

PROPOSED METHOD

1. Data sources

Two sources of information were needed for our study: (1) new sentiments and (2) historical prices.

Ten years tick (a tick is a measure of the minimum upward or downward movement in the price) and news data were collected from Reuters platform from January-01-2008 to December-31-2017 for four different stocks: AAPL for shares of apple company, GOOGL for Google shares, AMZN for Amazon shares, FB for Facebook shares.

Tick data was collected to include the following details: open bid, closing bid, high bid, and low bid, in addition to the time stamp. This high frequency data was collected to do intra-day short-term prediction. Our model requires at least one tick to be released every 1 hour, since we group our data hourly. This huge data requires some preprocessing that takes into consideration the big volume of data ($7 \text{ trading hours} * 3600 = 25200$ tick price per day) and the difference in interval between tick data. Tick data might have multiple prices released at the same second and miss some ticks at other seconds.

In addition to tick data we collected news sentiments. News data includes the stock symbol, date and time issued, source, title headline, sentiment (0 for neutral news, 1 for positive news and -1 for negative news), polarity of negative sentiment, polarity of

positive sentiment and polarity of neutral sentiment. The polarity of news is based on the count of positive/negative words in the news article.

2. Data pre-processing

Due to the huge amount of Tick data and to ease the manipulation of data, we imported our data to MySQL database where sorting data is done when querying.

The initial step was to replace missing ticks. Tick data have different time intervals in the data collected between ticks, such as a second might have four prices recorded while other seconds might not have a single price recorded. To fill missing ticks, we looked for the nearest tick data to fill our missing seconds. After importing data to our database and fill missing ticks, we grouped our data into one-minute time intervals where we would get the last tick received for each minute recorded in our data. Then we stored clean one-minute data in a new table (no weekends, no ticks outside market open time).

3. Align news with tick data

Unlike other approaches that filter news outside trading hours and news released during same intervals - so they overlap - we built different scenarios to try to take into consideration these cases. When generating our data, we gave the user an option to choose between one of the following three news scenarios:

- Last sentiment received on that day based on time to be used: for example, if we want to get the sentiment for 01-03-2010 at 14:00, we get the last sentiment received on 01-03-2010 before 14:00 and adopt it. If no sentiments exist, we consider the sentiment as neutral.

- Last sentiment during selected interval of time: if we are grouping our data into hourly time frame, we check the last sentiment released during this hour and consider it as dominant sentiment and if no news released, we consider the sentiment as neutral.

- Overall average for the day during selected interval: if more than one sentiment is released during the timeframe, we calculate the average for positive (a_p), negative (a_n) and neutral (a_{nu}) news (i.e:

$$a_p = \frac{\text{sum}(\text{positive news})}{\text{count}(\text{all news})}$$

). In case of equal sentiments, we sum the polarity of sentiments (polarity of positive sentiment, polarity of negative sentiment, polarity of neutral sentiment features) and check which of these features have the highest summation and consider it the dominant sentiment. In this scenario, we assume that weekend news is for Monday.

As for the tick data, we also gave the user an option to choose between two different scenarios:

- Generate data features from our one-minute database table based on hour interval, that is the input will be hourly data features, and the output, the next hour end price.

- Generate data features from our one-minute database table based on hour interval, that is the input will be hour data features, and the output, the last tick received that day upon closing of the market.

4. Features generation

Window sizes are tested in our models, i.e how many hours you wish to go back when you want to train the models. This generates our input data in the following format (window size * features).

The features used in our models are illustrated as follows:

- Maximum: Maximum price received during the selected hour
- Minimum: Minimum price received during the selected hour
- Average: Average price received during the selected hour

$$\frac{\sum p_i}{count(p_i)}$$

- Standard Deviation: Standard deviation of prices received during the selected hour

$$\sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2} \quad \text{where} \quad \mu = \frac{\sum p_i}{count(p_i)}$$

- Pseudo Log Return: logarithmic difference between average prices of two consecutive hours.

$$\ln\left(\frac{p_t}{p_{t-1}}\right) \quad \text{where } p_t \text{ is the average price at time } t$$

- Trend indicator: slope of linear model applied on tick data of the respective hour, which gives an idea about the trend during the last hour.
- Price: Last tick received at selected hour
- Sentiment: News sentiment analysis calculated based on chosen scenario.

So, our input data has 8 features and 1 output, the formula for the number of features is the following:

Features=8n where n is window size

The output of our model is end of day price for regression models and variation between today end of day price and yesterday end of day price for classification models.

5. Data Normalization

Since the features extracted from the input data are of different units and scale, normalization is needed to scale the data between 0 and 1, which also helps in faster convergence. To normalize our data, we use the minmaxscaler function provided by scikit-learn framework. This function gets the max and the min values of each column and apply the following formula

$$\frac{x_i - \min(x)}{\max(x) - \min(x)}$$

This normalization function was applied to our data - train data only - and then we applied normalization to our test data so that we can have all input features between 0 and 1.

Next, we experimented with various models, mainly: Recurrent Neural Network, Deep Neural Network, Support Vector Machine and Support Vector Regression.

6. Deep Neural Network

A neural network is a computational learning system, that translates and understands input data using network function to form desired output. Neural networks are widely used in supervised learning and reinforcement learning problems. So, an Artificial neural network is a software implementation of our brain neuronal structure where learning occurs by activating neural connections.

ANN can be trained on supervised and unsupervised data, where in supervised ANN, a labeled data set are given (an input with matching output). However, in unsupervised learning, the network attempts to understand input data on its own.

An Artificial neural network is based on a set of layers connected to each other where each layer is composed of neurons. Each neuron might take multiple weighted inputs, then apply the activation function and produce the output.

Neurons are basic units of neural network, a neuron take inputs, do calculations and produces output.

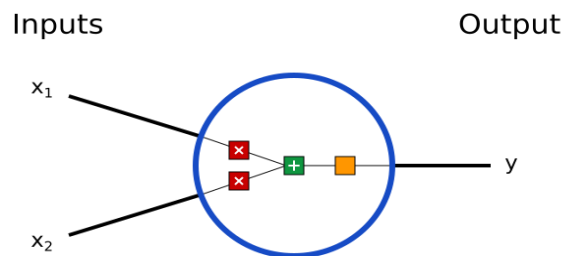


Figure 1: Neuron Structure

At each neuron, the input is multiplied by a weight and then weighted inputs are added with a bias b

$$(x_1 * w_1) + (x_2 * w_2) + b$$

Then activation function is applied

$$y = f(x_1 * w_1 + x_2 * w_2 + b)$$

The activation function is used to form predictable output of unbounded input.

So, a neural network is just a set of neurons connected to each other.

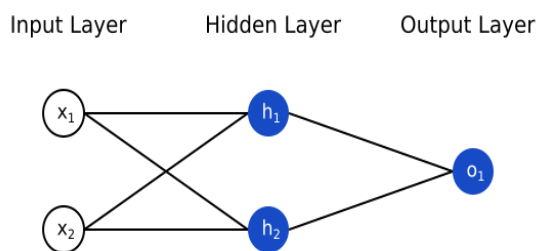


Figure 2: Neural network structure

This network has 2 inputs, 1 output and 1 hidden layer with 2 neurons (hidden layers are the layers located between input and output layers).

A deep neural network (DNN) is an artificial neural network (ANN) with multiple hidden layers between the input and output layers resulting in a certain level of complexity for the network.

We have different types of deep neural networks such as Deep Convolutional Neural Network (DCN), Deep Belief (DBN) Neural Network, Deep Feed Forward (DFF) Neural Network, Recurrent Neural Network (RNN), Long/Short Term Memory (LSTM) Neural Network, Gated Recurrent Unit (GRU) Neural Network and many others.

In our study, we focused on networks that fit with our problem and input features such as DNN, LSTM, RNN and GRU.

DNN have been widely used in different applications majorly in financial operations, trading business analytics and product maintenance. The types we chose to use in our study show a good performance for nonlinear and sequential data, as some of them have memory that can relate between inputs given it treats them as a sequence of events happening in chronological order.

- Deep Neural Network (DNN): DNN is the simplest form of neural network; in this network the data passes through different network nodes - in one direction - from first tier onwards to reach the output node.

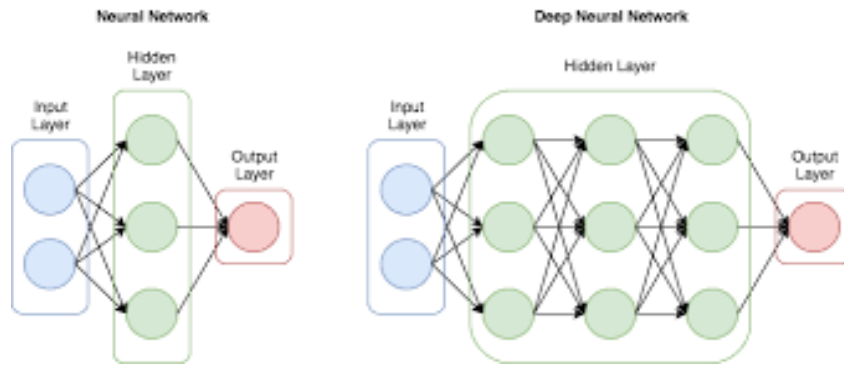


Figure 3: Deep Neural Network Structure

The structure of this model was applied based on related work [3], where we applied the same structure of the network, the input data was the features provided above, and the output was the actual price at the end of day with varying window sizes; in our study we took the first 4, 5 or 6 trading hours data features as 1 input and predicted end of day price.

The network was formed of 5 layers each with I, 4I/5, 3I/5, 2I/5, I/5 and 1 neuron where I represents our number of inputs. *Tanh* was the activation function used for all hidden layers and *linear function* for output layer.

H2O platform is a leading open source data science platform that is used to apply this network. The platform includes the implementation of deep learning algorithms; after splitting the data into 85% training and 15% testing, we trained the model for 50 epochs and applied ADADELTA [10] (adaptive learning rate algorithm) optimization algorithm to improve learning rate learning process [11]. It is a per-dimension adaptive learning rate method for gradient descent, not necessary to search parameters for gradient descent manually.

7. Recurrent Neural Network

Recurrent neural network is a member of the feed-forward neural network family; however, they are different in their ability to send information over time-steps through adding loops to the network. RNN is considered a good practice for time-dependent data due to the memory it has.

An RNN has a feedback loop that it uses to capture relationships between the ordered input data given. So, an RNN has an extra parameter “time-steps” that is used to create relationships between sequential data, by predicting the output not only based on the current input but also based on input at all previous time steps. The output of a layer is saved and fed back to the input layer.

The first layer is just the same as the ANN layer; that is, the product of the sum of the weights and features. The RNN process then begins in the subsequent layers. At each time-step, each node acts as a memory cell that will remember information from a previous time-step - that it may use later.

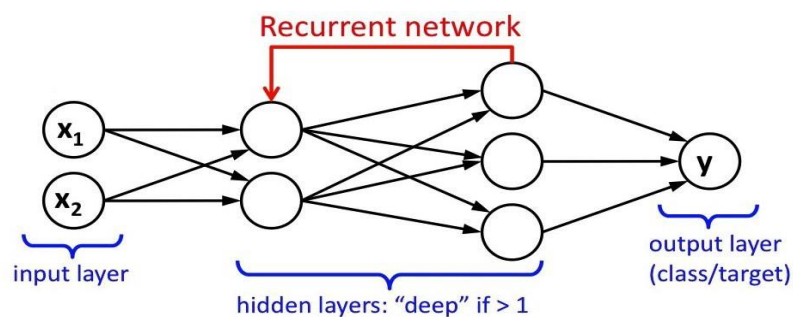


Figure 4: Simple RNN structure

A recurrent neural network has different cells that use different computations, the main three cell types and calculations can be illustrated below:

A general RNN network structure would have the input vector and previous hidden state as input, and current hidden state and output as productions.

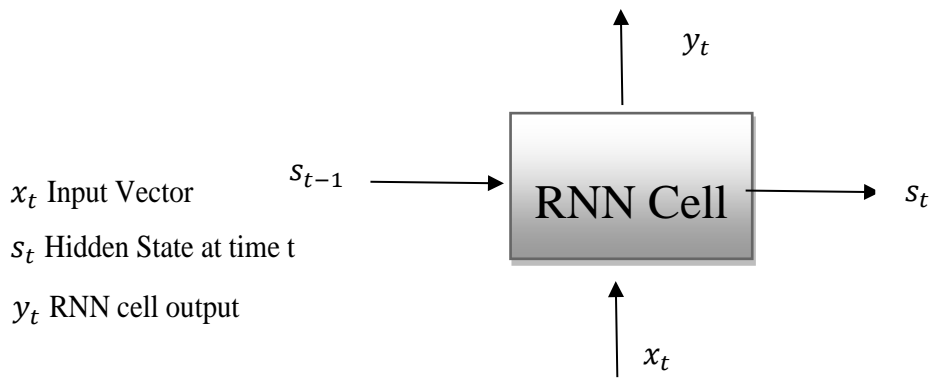


Figure 5: RNN Cell

$$s_t = f_w(s_{t-1}x_t)$$

- Vanilla RNN: This is the most basic RNN computational cell that can be, it has simple implementation, and a linear model followed by an activation function.

$$s_t = f_w(s_{t-1}x_t)$$

$$\downarrow$$

$$s_t = \tanh(W_{ss}s_{t-1} + W_{xs}x_t)$$

$$y_t = W_{sy}s_t$$

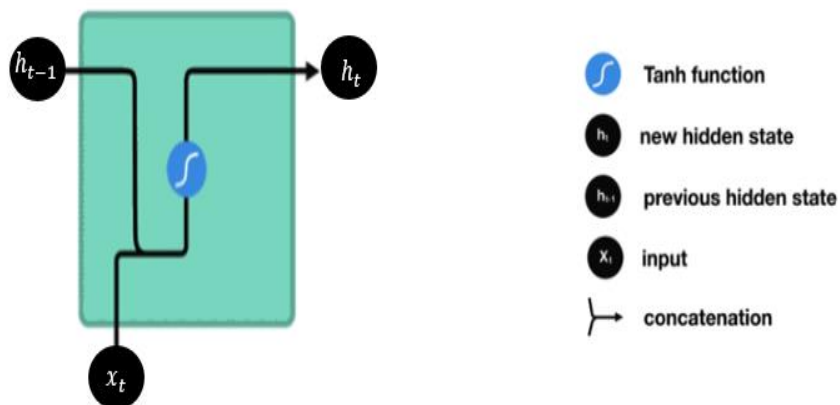


Figure 6: Vanilla RNN cell

First, the input and previous hidden states are combined to form a vector. That vector now has information on current and previous inputs. The vector goes through the *tanh* activation, and the output is the new hidden state, or the memory of the network.

Tanh function is used as the activation function to regulate values flowing through the network. *Tanh* function always produces a value between 0 and 1.

Vanilla RNN uses less computations than LSTM and GRU cells.

- Long-Short Term Memory (LSTM): LSTM rises due to the vanishing problem in vanilla RNN and its short-term memory. This cell has an internal mechanism called gates that can regulate the flow of information. These gates learn what data in a sequence is important to keep for the following state and what information to throw away.

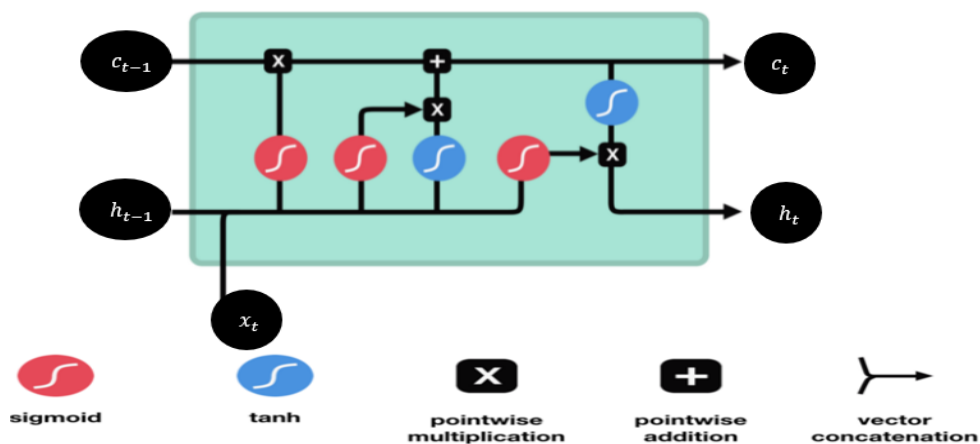


Figure 7: LSTM Cell

The LSTM cell consists of four gates: Cell Gate which can be considered as the memory of the network, Forget Gate, Input Gate and Output Gate.

Similarly to *tanh* used in vanilla RNN, LSTM uses a sigmoid activation function, that squishes the value between 0 and 1 instead of -1 and 1. This function would be helpful

in order to know what information to forget and what to update (a number multiplied by 0 will be forgotten and a number multiplied by 1 will be kept for the next state).

Forget Gate: decides what information should be kept and what should be discarded. It requires two information to be passed - information from the previous hidden state and current input passed to a sigmoid function – in order to produce a number between 0 and 1; the closer the number is to 0 means forget, and the closer to 1 means to keep.

Input Gate: is responsible for updating the cell state; it requires the previous hidden state and current input as information. This information passes over the sigmoid function producing a number between 0 and 1; we also pass this information over a *tanh* function to produce values between -1 and 1, which helps regulate the network. Then multiply the *tanh* output with the sigmoid output.

Cell Gate: calculates the cell state after having Forget Gate output and Input Gate output, first a pointwise multiplication between previous cell state and *Forget Gate* output, this will drop values in cell state if it is multiplied by 0, then pointwise addition applied on output of *Input Gate* output and previous calculated output, this will be the new cell state.

Output Gate: this state decides upon the next hidden state; the hidden state is also used to calculate the output. First, we pass previous hidden state and current input over a sigmoid function, then we pass the new cell state over a tanh function and finally multiply the *tanh* output with the sigmoid output to decide the new hidden state. The output is the hidden state and new cell state, and the new hidden state is fed back to the network in the next time step.

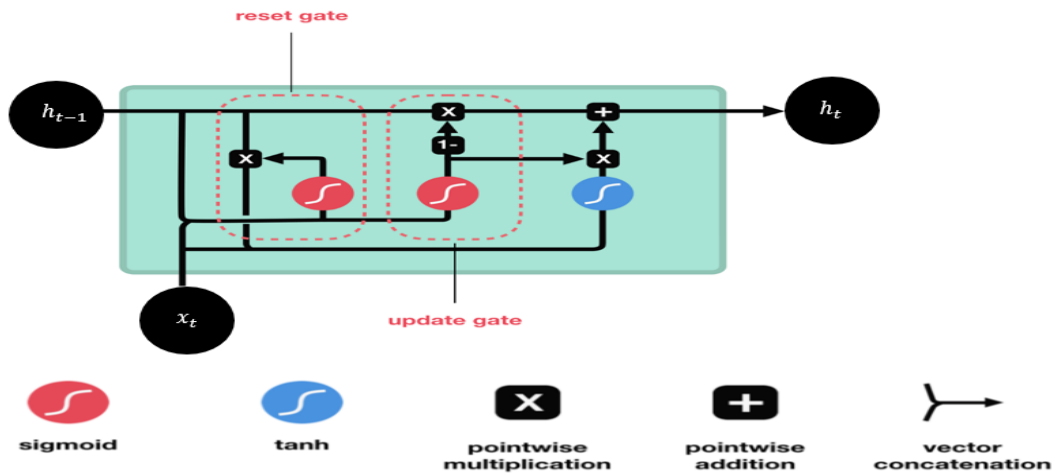


Figure 8: GRU Cell

People find LSTMs work well, but unnecessarily complicated, so they introduced GRUs. So, GRU is the newer version of RNN that is somehow like LSTM.

GRU get rid of cell state used in LSTM and used hidden state to transform information. So, it only has two gates, the reset gate and the update gate.

Update Gate: The update gate has the same job as *Forget Gate* in LSTM, it decides what information to keep and what to discard.

Reset Gate: This gate decides how much information should be passed from previous information.

So, to summarize things up, RNN is used for sequence data, vanilla RNN for short term memory, and LSTM and GRU were created to enhance the short-term memory of RNN using gates mechanism.

In our RNN model, we tried different network structures with different numbers of neurons at each layer. We tried 7, 6, 5, 4- and 3-layers networks, varying the number of neurons at each layer between 250 and 5 neurons, and tested the implemented networks to get the best results for 3- and 4-layers networks.

We trained and tested this model on 10 years data; the input data was the features provided above and the output is the actual price at the end of day depending on our window size. In our study, we took the first 4, 5 or 6 trading hours data features time step in the RNN and predict the end of day price.

Moreover, we tried different cells provided by TensorFlow given it was the platform. We used it to train RNN, so we have a trained model on Basic RNN cell, LSTM cell and GRU cell supported by the platform.

We split our data into 10% testing and 90% training and then we trained the model for 100 epochs and applied *ADAMOptimizer* as our optimization algorithm to get the best learning rate for our model.

8. Support Vector Machine

Support Vector Machine is a discriminative classifier that requires a labeled training dataset, and outputs an optimal hyperplane which categorizes new examples.

This supervised machine learning algorithm can be used for both regression and classification problems. This algorithm uses a kernel trick technique that transforms the data and then finds the optimal boundary between outputs. Moreover, SVM shows that it can perform well on non-linear dataset problems, based on the kernel we choose in training the SVM model.

Learning of the optimal hyperplane is done by transforming the problem thanks to some linear algebra with the largest margin; the margin between the classes represents the longest distance between the closest data points of those classes. The larger the

margin, the lower the error. This process is done using a kernel, of which there are different types:

- *Linear Kernel*: a normal dot product

$$k(x_i, x_j) = (x_i \cdot x_j)$$

- *Polynomial Kernel*: generates new features by applying polynomial combination of existing features.

$$k(x_i, x_j) = (x_i \cdot x_j + a)^d$$

d is the degree of the polynomial, a is a constant term.

- *RBF Kernel*: is used when there is no prior knowledge about the data. It generates new features based on measuring distance between all data points with respect to a specific point.

$$k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$$

$$\gamma > 0$$

we have 2 main parameters in this model: c is the regularization parameter and γ defines how far the influence of a single training reaches. As c or γ increase, the model gets overfit, and as c or γ decrease, the model gets underfit.

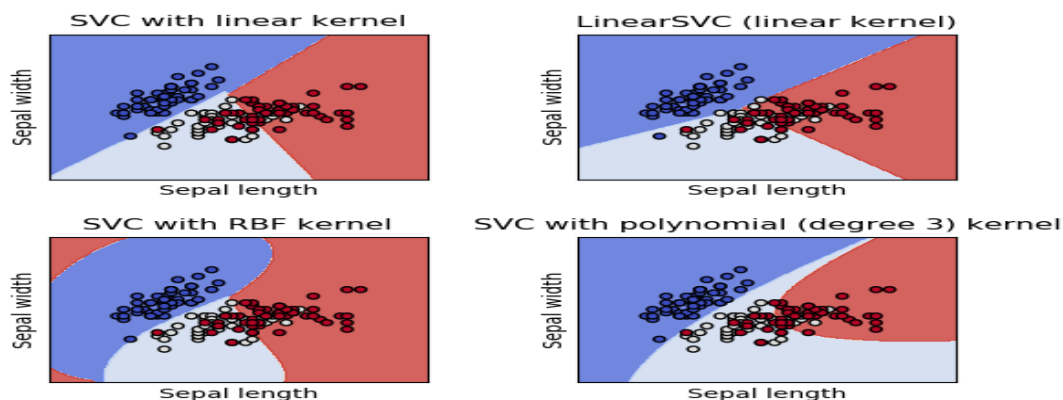


Figure 9: Examples of SVM

In our SVM model, we tried different kernel algorithms, tuning parameters for each model, Linear, Polynomial and RBF.

We trained and tested this model on 10 years data; the input data is the features provided above and the output is the binary - either 0 or 1; 1 when the today end price goes down with respect to yesterday end of day price, and 0 when the price goes up. We varied our window size; in our study we took the first 4, 5 or 6 trading hours data features.

We used sickit learn library to build this model and split our data into 10% testing and 90% training, and then we trained the model and applied GridSearchCV to choose the best parameters to fit our model.

9. Support Vector Regression

Support vector Regression is the same as SVM; given its name, we can know that it is used for regression instead of classification. It uses the same terms and functionalities as SVM to predict continuous value. In SVR we try to fit the error within a defined threshold unlike other regression algorithms that try to minimize the error between predicted and actual values.

In this model we follow the same process of SVM except for the out, the output in this model is not a class, it is the actual end of day price.

CHAPTER 4

RESULTS AND DISCUSSION

In this section, we will discuss the different results we achieved for the models defined in the above section; both news sentiments and stock prices serve as data sets for our experiment.

1. Data Description

Below is a description of the data we had before applying our models:

Table 1: Data Description

Stock Name	Data in Years	Trading Days	Total Data Points	Total Articles	EOD Positive/Negative Output Direction
AAPL	2008 - 2018	2749	19,243	78036	1478 positives 1271 negatives
FB	2012 - 2018	1645	11,515	30198	886 positives 759 negatives
GOOGL	2014 - 2018	1175	8,225	19829	625 positives 550 negatives
AMZN	2008 - 2018	2749	19,243	37265	1450 positives 1299 negatives

Data in Years: the years for which we have data, for different stocks, for both news articles and stock market tick data.

Trading Days: is the number of trading days for the selected years, without holidays, weekends and closed-market days.

Total Data Points: for our hourly-grouped data features (1 data point for each hour) and 7 trading hours per day - from 9:30 to 16:00.

Total Data Points = Trading Days * 7 Trading Hours

Total Articles: is the number of articles we have for each stock during the years selected

EOD Positive/Negative Output Direction: is the number of positive and negative outputs when calculating directional accuracy for our model:

$$\text{Output Direction} = \begin{cases} 1, & p_{eod} - p_{eod-1} < 0 \\ 0, & p_{eod} - p_{eod-1} \geq 0 \end{cases}$$

Where 1 represents that today's end-of-day price (p_{eod}) is going down with respect to yesterday's end-of-day (p_{eod-1}) price, and 0 represents that the price is going up. So, the positive output direction is the count of 0's in our data set, and negative output direction is the count of 1's.

For all the models we tested (RNN, SVM, SVR), we split our data into 90% training and 10% testing. As for our DNN model, we split our data into 85% training and 15% testing to compare our results-related work model [3].

Below are the figures of stock price movements over time:

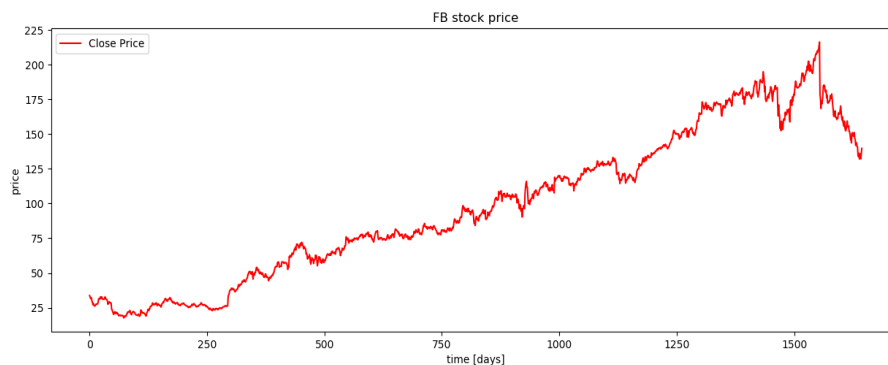


Figure 10: FB Stock Price

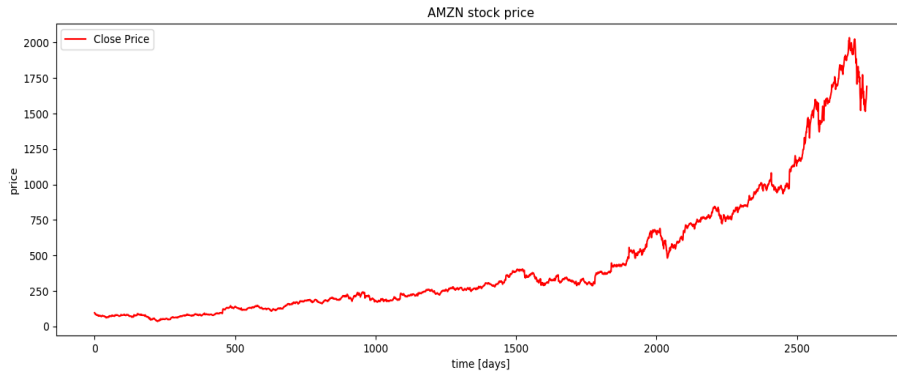


Figure 11: AMZN Stock Price

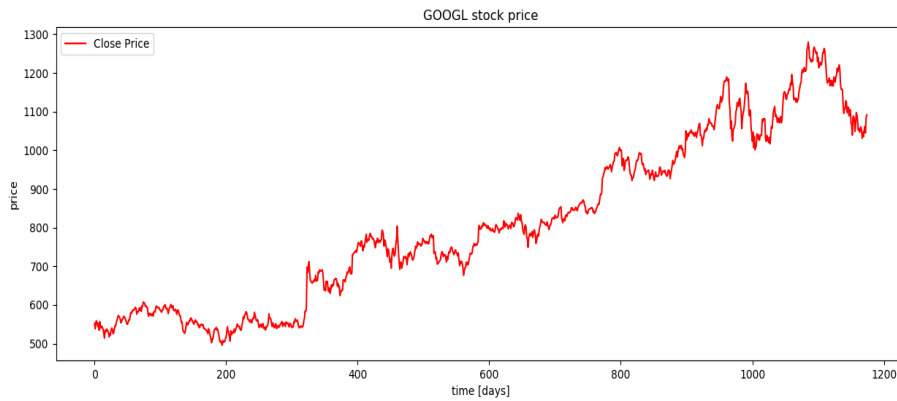


Figure 12: GOOGL Stock Price

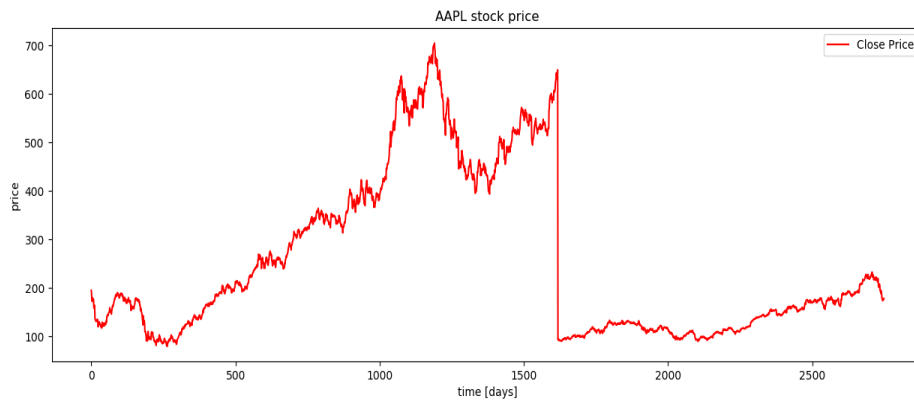


Figure 13: AAPL Stock Price

We wish to note that the sudden move in APPL stock, from a price of \$645.57 to \$92.44, is due to “stock split”, where APPL company issued more shares to existing investors in order to bring the price down - current shareholders got 7 new shares of

AAPL stock for each stock they previously owned. So, the new price is 1/7 of the previous price.

2. RNN Results

We applied our RNN network on the TensorFlow platform, after normalizing data using *minmaxscaler* and splitting our data into training and testing data frames, and trained our network for 100 epochs; this was selected based on trial and error-based MSE, so that we could figure out the best number of epochs by monitoring mean-squared error. As long as MSE decreases, you can increase the number of epochs, and once MSE start to increase, you should know that you have reached the best number of epochs.

As for the parameter used for RNN “time-steps” that represents window size, we tried 3 different time steps 4,5 and 6 - where for time-step=4, our data represented the first 4 trading hours for each day and end of day price as output. We tried the different window sizes with 3 different sentiment scenarios described in the earlier section. For the learning rate, we chose 0.0001 as our initial value and applied *ADAMoptimizer* to optimize our learning rate.

Also, a parameter we used for RNN is *batch size*, which is the number of samples processed before the model is updated, where $1 < \text{batch size} < \text{training set}$; for our model we chose an arbitrary batch size= 25.

The input in RNN TensorFlow is called tensor, that is a 3-dimension input [batch size, time-step, features], and the same goes for output which is a tensor of the following format [batch size, time-step, output size].

We tried a different number of layers and neurons to get the best results for a 3-layer network with 150, 100, 50 neurons respectively. During our test, we found that the shallower the network, the better the performance. As for the activation function, we used ReLU (Rectified Linear Unit) activation function for all layers; this function is the most used function in the neural network.

$$\text{ReLU} = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

This model with the above parameter was applied for different RNN cells, and supported by TensorFlow: basic LSTM, LSTM, basic RNN and GRU.

Different metric measures were used to evaluate this model such as Mean-squared-error (MSE), and directional accuracy.

Directional Accuracy Metric:

$$p_{dir} = \begin{cases} 0, & p_{eod} - p_{eod-1} < 0 \\ 1, & p_{eod} - p_{eod-1} \geq 0 \end{cases} \quad p_{eod} = \text{Predicted end of price at day } d$$

$$\text{Directional Accuracy} = \frac{\text{count}(p_{dir}=p_{act})}{\text{count}(d)}$$

p_{act} same as p_{dir} for actual price instead of predicted

$$\text{MSE} = \sum_1^n \frac{(p_{predicted} - p_{actual})^2}{n}$$

Table 2: AAPL RNN Results

Sentiment-Window	GRU		LSTM		Basic LSTM		Basic RNN	
	Directional Accuracy	MSE	Directional Accuracy	MSE	Directional Accuracy	Directional Accuracy	MSE	Directional Accuracy
S1-4	69.34%	0.00074	67.88%	0.00114	S1-4	69.34%	0.00074	67.88%
S1-5	68.25%	0.00039	69.34%	0.00064	S1-5	68.25%	0.00039	69.34%
S1-6	69.71%	0.00046	70.44%	0.00068	S1-6	69.71%	0.00046	70.44%
S2-4	68.98%	0.00012	66.42%	0.00063	S2-4	68.98%	0.00012	66.42%
S2-5	71.17%	0.00041	68.61%	0.00011	S2-5	71.17%	0.00041	68.61%
S2-6	71.53%	0.00073	72.63%	0.00014	S2-6	71.53%	0.00073	72.63%

S3-4	68.98%	0.00011	68.25%	0.00065	S3-4	68.98%	0.00011	68.25%
S3-5	69.71%	0.0002	68.98%	0.00071	S3-5	69.71%	0.0002	68.98%
S3-6	71.53%	0.00018	71.53%	0.00024	S3-6	71.53%	0.00018	71.53%

Table 3: AMZN RNN Results

Sentiment-Window	GRU		LSTM		Basic LSTM		Basic RNN	
	Directional Accuracy	MSE	Directional Accuracy	MSE		Directional Accuracy	MSE	Directional Accuracy
S1-4	71.53%	0.000352	69.34%	0.000425	S1-4	71.53%	0.000352	69.34%
S1-5	73.36%	0.000828	68.25%	0.000300	S1-5	73.36%	0.000828	68.25%
S1-6	74.47%	0.000249	57.3%	0.000234	S1-6	74.47%	0.000249	57.3%
S2-4	65.33%	0.000752	56.2%	0.000646	S2-4	65.33%	0.000752	56.2%
S2-5	66.22%	0.000438	57.2%	0.000471	S2-5	66.22%	0.000438	57.2%
S2-6	74.47%	0.000080	58.39%	0.000385	S2-6	74.47%	0.000080	58.39%
S3-4	61.68%	0.000826	51.46%	0.000714	S3-4	61.68%	0.000826	51.46%
S3-5	73.55%	0.000397	57.3%	0.000541	S3-5	73.55%	0.000397	57.3%
S3-6	74.1%	0.000244	52.92%	0.000350	S3-6	74.1%	0.000244	52.92%

Table 4: FB RNN Results

Sentiment-Window	GRU		LSTM		Basic LSTM		Basic RNN	
	Directional Accuracy	MSE	Directional Accuracy	MSE		Directional Accuracy	MSE	Directional Accuracy
S1-4	64.42%	0.000924	65.64%	0.000375	S1-4	64.42%	0.000924	65.64%
S1-5	66.87%	0.000231	67.48%	0.000502	S1-5	66.87%	0.000231	67.48%
S1-6	68.1%	0.000446	68.1%	0.000621	S1-6	68.1%	0.000446	68.1%
S2-4	64.42%	0.000626	63.8%	0.000470	S2-4	64.42%	0.000626	63.8%
S2-5	63.8%	0.000749	67.48%	0.002057	S2-5	63.8%	0.000749	67.48%
S2-6	68.71%	0.000164	69.33%	0.000252	S2-6	68.71%	0.000164	69.33%
S3-4	64.42%	0.000621	65.64%	0.001079	S3-4	64.42%	0.000621	65.64%
S3-5	67.48%	0.000187	66.26%	0.000390	S3-5	67.48%	0.000187	66.26%

S3-6	61.11%	0.000123	60.49%	0.000440	S3-6	61.11%	0.000123	60.49%
------	--------	----------	--------	----------	------	--------	----------	--------

Table 5: GOOGL RNN Results

Sentiment-Window	GRU		LSTM		Basic LSTM		Basic RNN	
	Directional Accuracy	MSE	Directional Accuracy	MSE		Directional Accuracy	MSE	Directional Accuracy
S1-4	60.68%	0.000426	61.54%	0.000311	S1-4	60.68%	0.000426	61.54%
S1-5	63.25%	0.000357	64.1%	0.000394	S1-5	63.25%	0.000357	64.1%
S1-6	63.25%	0.000303	65.81%	0.000485	S1-6	63.25%	0.000303	65.81%
S2-4	60.68%	0.000260	62.39%	0.000352	S2-4	60.68%	0.000260	62.39%
S2-5	62.39%	0.000474	64.1%	0.000447	S2-5	62.39%	0.000474	64.1%
S2-6	63.25%	0.000392	65.81%	0.000478	S2-6	63.25%	0.000392	65.81%
S3-4	60.68%	0.000374	62.39%	0.000319	S3-4	60.68%	0.000374	62.39%
S3-5	63.25%	0.000427	64.10%	0.001198	S3-5	63.25%	0.000427	64.10%
S3-6	65.81%	0.000436	64.96%	0.000328	S3-6	65.81%	0.000436	64.96%

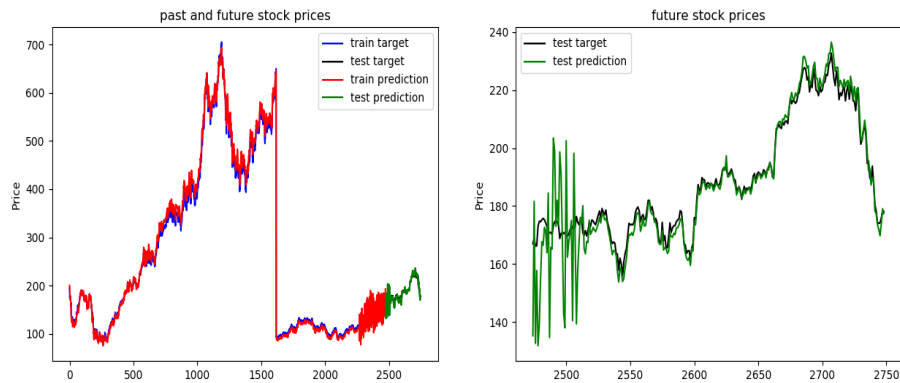


Figure 14: AAPL RNN Stock Price Prediction

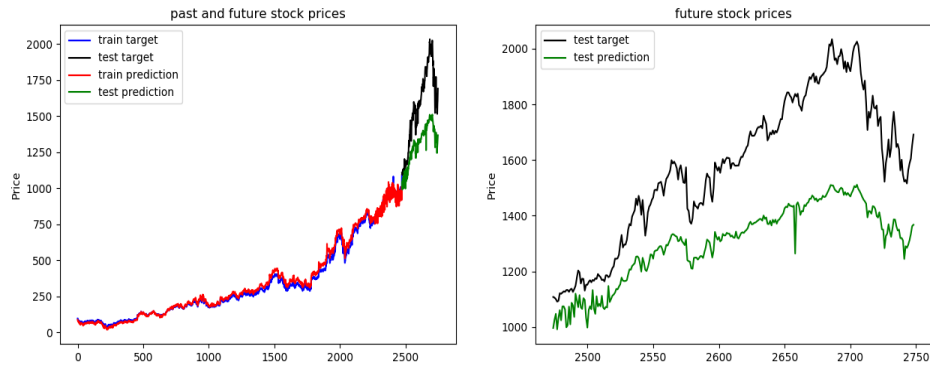


Figure 15: AMZN RNN Stock Price Prediction

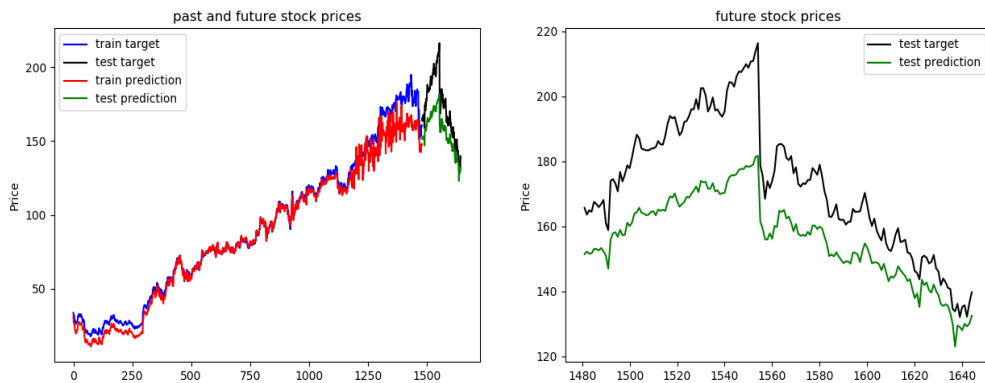
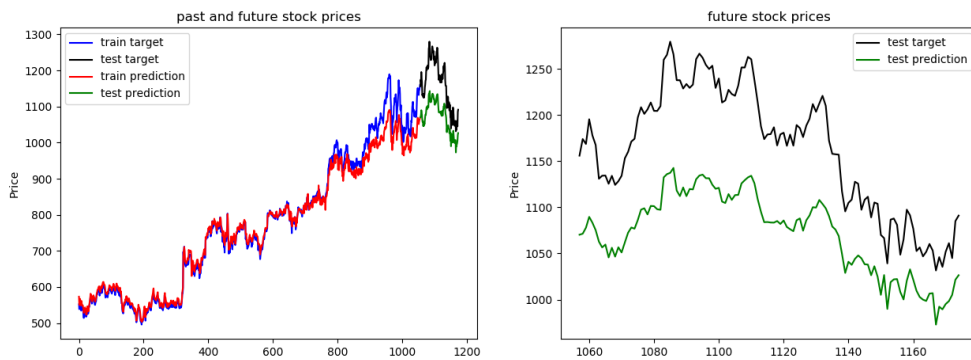


Figure 16: FB RNN Stock Price Prediction

Figure



17: GOOGL RNN Stock Price Prediction

From the above results, we distinguish that basic RNN (vanilla RNN) shows the best performance in terms of accuracy for different stocks. Also, we can see that “different sentiments” results in the best accuracy for different stocks, which might be due to different news effects on different stocks. As for the window size, taking the first 6 trading

hours of the day and prediction of the end-of-day price shows the best performance among all stocks.

The directional accuracy calculated shows a value between 68% and 80% for different stocks, and from our graphs, we can see that the price prediction is not as accurate as the direction is.

We would like to note here, that we tried intraday prediction (predicting the next-end-hour price) based on different window sizes and that wasn't accurate for our model formulation given that the maximum accuracy we reached was 59%. So, we went with the other direction which is the end-of-day price.

For the best models, we also calculated the True Negative Values (TN), True Positive Values (TP), False Negative Values (FN) and False Positive Values.

Table 6: RNN Metrics

Stock	FP	TP	FN	TN	Precision	Recall	f-Measure
AAPL	28	120	23	103	0.81	0.84	0.82
AMZN	34	126	22	92	0.78	0.85	0.81
GOOGL	20	40	17	40	0.67	0.7	0.68
FB	24	58	21	60	0.7	0.73	0.71

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

$$\text{f-measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

3. DNN Results

We applied our DNN network on the H2O platform, divided our data into 85% training and 15% testing data frames, and trained our network for 50 epochs. Parameters for this model were selected in order to compare our model with the features of another

related work presented earlier [3]. As for the parameters used for the DNN ADADELTA optimization algorithm using $\rho = 0.9999$ and $\epsilon = 10^{-10}$, we tried our model for 3 different window sizes 4,5 and 6 where for window size=4 our data represented the first 4 trading hours for each day and the end-of-day price as output. We tried these different window sizes with 3 different sentiment scenarios described in an earlier section.

The input in the DNN model was the first $n * 8$, where n is the window size, 8 is the number of features, and 1 output which is the end-of-day price.

Three different networks were tried based on window size with I, 4I/5, 3I/5, 2I/5, I/5 and 1 neuron; I being the number of inputs + output. As for the activation function, *tanh* was the function used for all layers. The best directional accuracy was achieved with window size 6, for different stocks.

Table 7: DNN Results

Sentiment-Window	AAPL	AMZN	GOOGL	FB
	Directional Accuracy			
S1-4	80.19%	73.75%	73.79%	71.89%
S1-5	79.41%	73.41%	75.86%	72.51%
S1-6	80.93%	74.03%	80.06%	72.68%
S2-4	75.78%	70%	77.52%	70%
S2-5	78.32%	73.12%	75.59%	70.63%
S2-6	81.32%	72.44%	80.10%	71.98%
S3-4	78.05%	68.68%	72.89%	72.26%
S3-5	80.88%	73.54%	77.83%	70.22%
S3-6	80.24%	73.43%	78.65%	71.84%

Overall, the network achieved a directional accuracy between 68% and 81%, knowing that in [3] the best directional accuracy achieved for AAPL stock was 66%. So,

grouping data into hours instead of minutes, and integrating news sentiment enhances the performance of the model.

Table 8: DNN Metrics

Stock	FP	TP	FN	TN	Precision	Recall	f-Measure
AAPL	37	138	42	206	0.78	0.76	0.77
AMZN	45	128	63	180	0.73	0.67	0.69
GOOGL	20	70	17	79	0.77	0.80	0.78
FB	30	73	35	100	0.7	0.67	0.68

4. SVR Model

We applied our SVR using the *sickit* learn library platform, after normalizing data using *minmaxscaler* and splitting our data into 90% training and 10% testing data frames.

During the training phase, parameters of the model were determined by grid search in 5-fold cross-validation. Different kernels were used in the experiment's RBF, linear and polynomial. For parameter C, we tried C=0.001, 0.01, 0.1,0.2,0.5,0.8 ,1, 10. For γ , we set $\gamma= 0.001, 0.01,0.5, 0.1, 1,2,3,4,5$ for the nonlinear search. For d (degree of the polynomial kernel), we had the following values d= 2,3,4,5,6,7,8. Thus, there were too many combinations of parameters at each time-point in total. For each combination of the parameters, 5-fold cross-validation was conducted to validate the trained model.

Applying cross validation on time series data required an approached picking, instead of randomly picking a certain subset from the set, but since our data were grouped into hourly features and each data entry represented a day, then a random picking cross-validation could be applied in that case. Based on the selected parameter from grid search,

the one with the best performance was used to configure the final model for independent testing.

With the selected parameters, we trained our network on a data sample with $n*8$ features, where n is the window size, 8 is the number of features we used, and 1 output which is the end-of-day price.

Directional accuracy was the metric selected to evaluate our model.

Directional Accuracy Metric:

$$p_{dir} = \begin{cases} 0, & p_{eod} - p_{eod-1} < 0 \\ 1, & p_{eod} - p_{eod-1} \geq 0 \end{cases} \quad p_{eod} = \text{Predicted end of price at day } d$$

$$\text{Directional Accuracy} = \frac{\text{countif}(p_{dir}=p_{act})}{\text{count}(d)}$$

p_{act} same as p_{dir} for actual price instead of predicted

Table 9: SVR AAPL Results

Sentiment-Window	AAPL Directional Accuracy		
	Polynomial	RBF	Linear
S1-4	72.63%	70.07%	72.26%
S1-5	75.91%	70.44%	71.9%
S1-6	78.83%	73.72%	78.47%
S2-4	71.17%	64.6%	58.18%
S2-5	75.55%	64.6%	57.82%
S2-6	78.83%	67.15%	58.18%
S3-4	72.63%	64.96%	72.26%
S3-5	74.82%	64.96%	71.9%
S3-6	79.2%	68.61%	78.47%

Table 10: SVR AMZN Results

Sentiment-Window	AMZN Directional Accuracy		
	Polynomial	RBF	Linear
S1-4	71.9%	70.44%	71.17%
S1-5	71.36%	70.07%	71.17%
S1-6	70.64%	71.53%	71.26%

S2-4	71.53%	71.53%	71.9%
S2-5	72.26%	71.9%	70.63%
S2-6	72.01%	70.45%	71.45%
S3-4	70.8%	69.71%	70.8%
S3-5	70.36%	71.26%	71.63%
S3-6	71.74%	70.09%	70.45%

Table 11: SVR FB Results

Sentiment-Window	FB Directional Accuracy		
	Polynomial	RBF	Linear
S1-4	62.07%	61.21%	61.21%
S1-5	64.66%	61.21%	62.07%
S1-6	63.83%	65.52%	61.38%
S2-4	62.07%	61.21%	61.21%
S2-5	66.38%	62.07%	61.21%
S2-6	64.83%	65.52%	65.52%
S3-4	62.07%	61.21%	61.21%
S3-5	65.38%	62.07%	61.21%
S3-6	65.83%	65.52%	65.52%

Table 12: SVR GOOGL Results

Sentiment-Window	GOOGL Directional Accuracy		
	Polynomial	RBF	Linear
S1-4	65.64%	65.64%	66.26%
S1-5	64.41%	67.48%	63.71%
S1-6	64.85%	67.94%	65.55%
S2-4	65.64%	65.64%	66.26%
S2-5	60.55%	67.71%	67.71%
S2-6	64.23%	65.94%	66.55%
S3-4	65.64%	65.64%	66.26%
S3-5	68.71%	65.21%	68.01%
S3-6	62.35%	63.58%	62.96%

As shown in the above results, using polynomial kernel has produced the best result in terms of directional accuracy for the different sticks we tested.

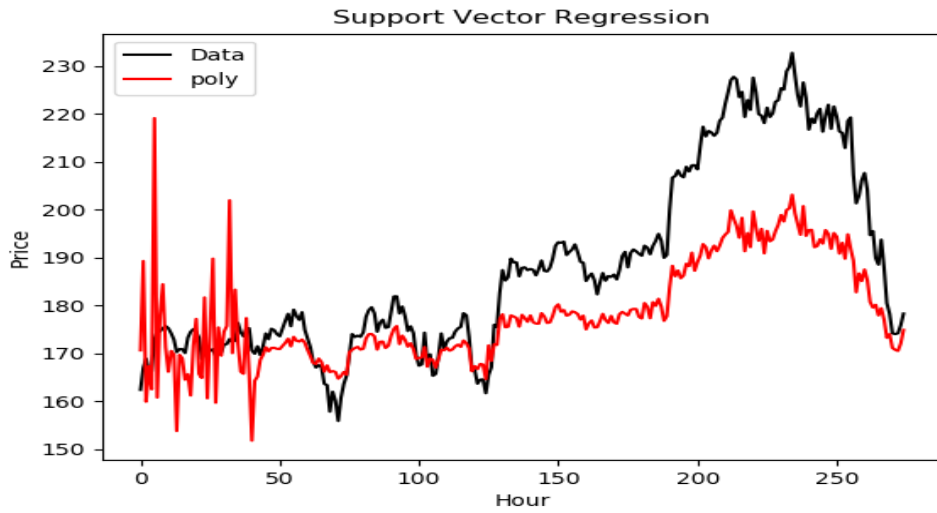


Figure 18: SVR AAPL Stock Price Prediction

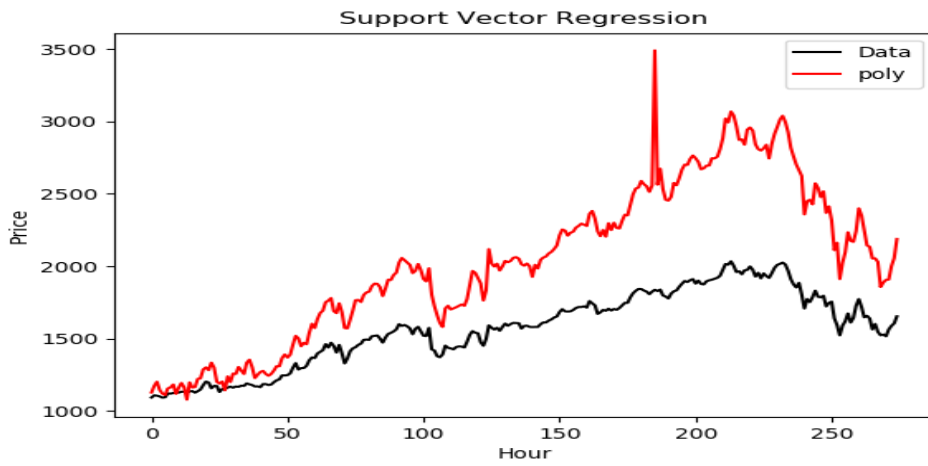


Figure 19: SVR AMZN Stock Price Prediction

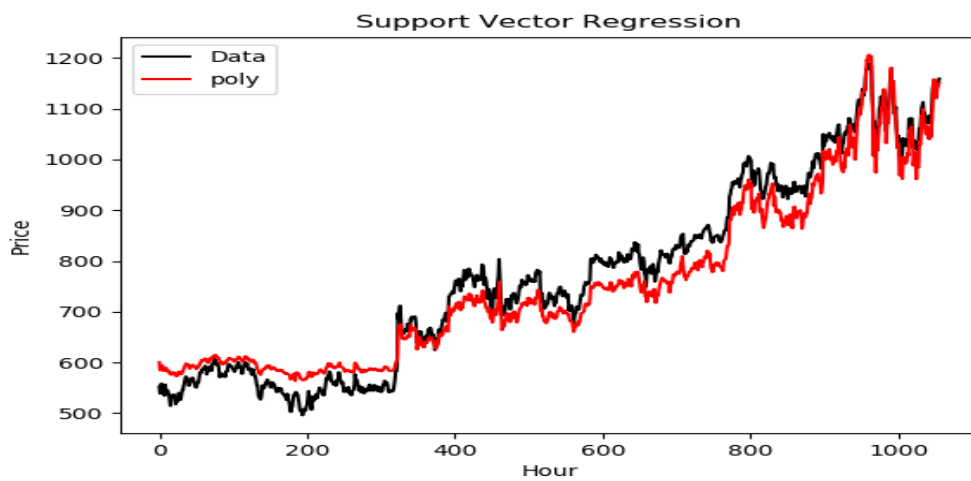


Figure 20: SVR GOOGL Stock Price Prediction

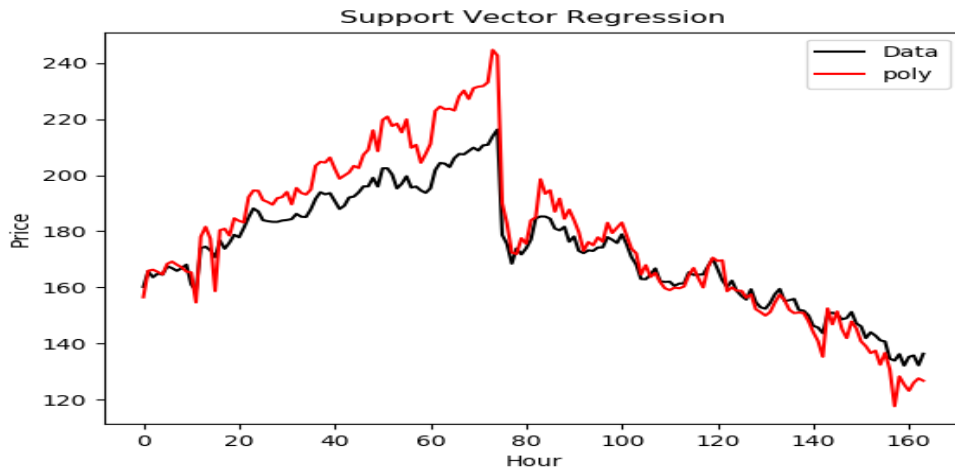


Figure 21: SVR FB Stock Price Prediction

Different parameter values were established for different stocks, based on search grid.

Table 13: SVR Metrics

Stock	FP	TP	FN	TN	Precision	Recall	f-Measure
AAPL	33	116	24	101	0.77	0.82	0.79
AMZN	42	118	34	80	0.73	0.77	0.74
GOOGL	21	38	18	39	0.64	0.67	0.65
FB	28	54	23	58	0.65	0.7	0.67

We would like to note here that we tried to calculate the accuracy for this model based on the last price received and end-of-day data; the best accuracy we got for this strategy was 60%.

5. SVM Results

We applied the same approach we followed in SVR, but with a different out feature calculated for this data frame. Since SVR is a regression model, it can predict real prices, however, in SVM we divide the data into classes. And in our case, we had 2 classes

0 and 1. 0 for end-of-day prices going down with respect to yesterday's end-of-day price, and 1 for going up end-of-day trends.

So, we formulated our new data frame with the newly calculated output and trained our model to get the following results based on the accuracy metric we chose:

Table 14: SVM APPL Results

Sentiment-Window	AAPL Directional Accuracy		
	Polynomial	RBF	Linear
S1-4	78.18%	54.55%	55.27%
S1-5	83.36%	61.09%	67.27%
S1-6	81.73%	64.73%	69.45%
S2-4	79.27%	66.18%	55.64%
S2-5	82.64%	61.45%	67.27%
S2-6	81.09%	65.45%	70.18%
S3-4	79.27%	66.91%	56%
S3-5	82.91%	62.18%	68.36%
S3-6	81.64%	71.27%	70.91%

Table 15: SVM AMZN Results

Sentiment-Window	AMZN Directional Accuracy		
	Polynomial	RBF	Linear
S1-4	75.27%	58.55%	58.55%
S1-5	74.91%	58.55%	66.55%
S1-6	65.82%	63.27%	75.27%
S2-4	74.18%	58.55%	58.55%
S2-5	74.18%	58.55%	65.82%
S2-6	68.36%	63.64%	75.27%
S3-4	75.64%	58.55%	58.55%
S3-5	70.18%	58.55%	66.18%
S3-6	68.73%	63.64%	73.82%

Table 16: SVM GOOGL Results

Sentiment-Window	GOOGL Directional Accuracy		
	Polynomial	RBF	Linear
S1-4	70.94%	50.43%	58.12%

S1-5	80.34%	57.26%	67.52%
S1-6	79.62%	69.23%	71.79%
S2-4	70.94%	50.43%	58.12%
S2-5	77.78%	54.7%	68.38%
S2-6	79.76%	64.96%	71.79%
S3-4	70.09%	50.43%	58.12%
S3-5	76.92%	55.56%	67.52%
S3-6	76.62%	65.81%	71.79%

Table 17: SVM FB Results

Sentiment-Window	FB Directional Accuracy		
	Polynomial	RBF	Linear
S1-4	68.9%	50.61%	54.88%
S1-5	73.17%	68.29%	67.68%
S1-6	74.66%	71.34%	73.78%
S2-4	73.17%	50.61%	54.88%
S2-5	74.01%	67.07%	67.07%
S2-6	73.27%	71.34%	73.17%
S3-4	75%	50.61%	54.88%
S3-5	73.78%	68.9%	66.46%
S3-6	60.74%	50.31%	62.58%

As shown in the above results, using polynomial kernel produced the best result in terms of directional accuracy for the different sticks we tested.

Table 18: SVM Metrics

Stock	FP	TP	FN	TN	Precision	Recall	f-Measure
AAPL	25	125	22	103	0.83	0.85	0.83
AMZN	46	115	22	92	0.71	0.83	0.77
GOOGL	14	45	9	49	0.76	0.83	0.8
FB	11	72	30	51	0.75	0.76	0.76

From the above models, we can distinguish that SVM shows the best performance in terms of accuracy and f-measure on our data features using a polynomial kernel with degree 2, all the while detecting other parameters using grid search.

To check how our model performs in real life, we proposed a trading strategy to evaluate our model and check how much we can gain using this model.

Table 19: Best model accuracy for each stock

	SVM	SVR	DNN	RNN
APPL	82.91%	79.2%	81.32%	81.3%
AMZN	75.27%	72.26%	74.03%	74.56%
GOOGL	80.34%	66.38%	80.1%	68.38%
FB	75%	68.71%	72.68%	72.39%

According to Table 14,15,16 and 17, it is very clear that our SVM model is able to achieve accuracies way above the 50%. When looking at Table 19, it also clear that SVM outperforms SVR, DNN, and RNN. All achieved accuracies are above 75% and in the case of APPL, the achieved accuracy is around 83%.

CHAPTER 5

PROPOSED STRATEGY

1. Risk Model

The results of each model were used with the following strategy: for each trading day, we can check if the current closing price - given our best window - is less than yesterday's closing price, and if our model predicts that the movement is up (yesterday's EOD with respect to today's EOD) then we buy a share of the stock.

On the other hand, if the current closing price is greater than yesterday's price and the model predicts a down-trend move, we sell the stock share. Otherwise, we don't do anything.

At the end of each day, we pretend to close the trade we took.

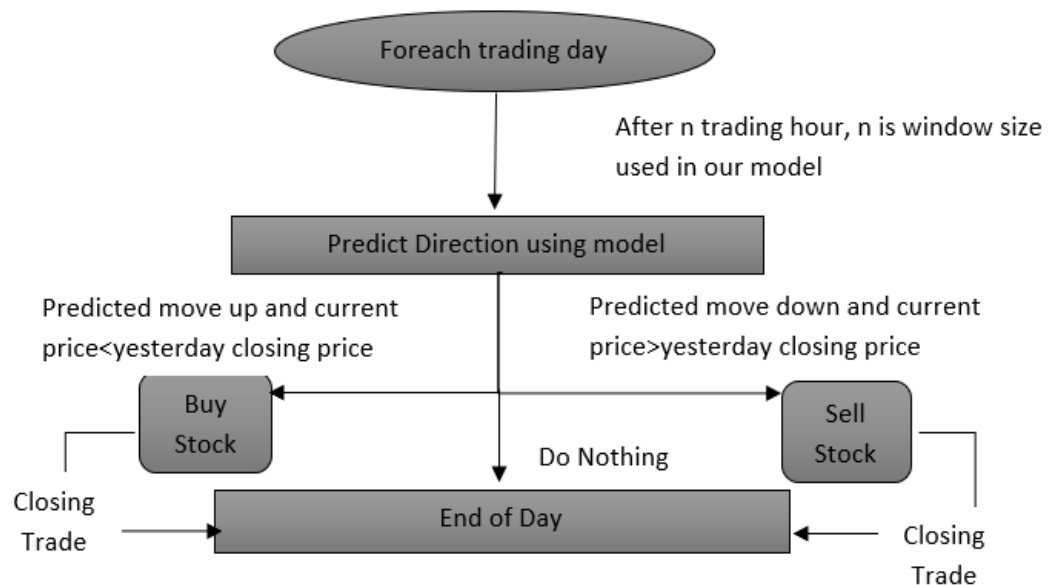


Figure 22: Proposed Risk Strategy

2. Risk Model Results

We also tried to take trades based on the predicted direction regardless of what the current price is with respect to yesterday's closing price. This might be more profitable, but the risk strategy is more secure.

Below are samples of the profits when applying the risk strategy and when not.

For the same testing data, we calculated the average price over a trading period to check how many shares we can buy/sell with a trading account balance of \$100,000.

- For AAPL share, the average price is \$190, so with \$100,000 balance we can buy 526 shares. So, in order to keep a margin of floating gain/loss, we bought/sold 500 stocks on each trade. We ran 273 trades in total, where 160 were winning and 113 were losing - without applying the risk strategy defined above; our total profit gained was \$35,850.00.

However, when we applied the risk strategy, only 37 trades were taken - 25 of which were winning - and 12 were losing, ending in \$14,465.00 as total profit.

- For AMZN share, the average price is \$1600, so with a \$100,000 balance we bought 62 shares, and in order to keep a margin of floating gain/loss, we bought/sold 60 stocks on each trade. We ran 273 trades in total, where 147 were winning and 126 were losing - without applying the risk strategy; our total profit gained was \$35,580.00. However, when we applied the risk strategy, only 47 trades were taken - 27 of which were winning and 19 were losing, ending with \$582.00 as total profit.

- For GOOGL share, the average price is \$1100, so with a \$100,000 balance we bought 91 shares; to keep a margin of floating gain/loss we bought/sold 85 stocks on each trade. We ran 116 trades in total, where 58 were winning and 58 were losing - without applying the risk strategy; our total profit gained was \$846. However, when we applied the risk strategy, only 14 trades were taken – 9 of which were winning and 5 were losing, ending with \$2,643.00 as total profit.

- For FB share, the average price is \$200, so with a \$100,000 balance we bought 500 shares; to keep a margin of floating gain/loss we bought/sold 490 stocks on each trade. We ran 163 trades in total, where 89 were winning and 73 were losing - without applying the risk strategy; total profit gained was \$16,072.00. However, when we applied the risk strategy, only 32 trades were taken - 22 of which were winning and 10 were losing, ending with \$4,851.00 as total profit.

Below is the risk strategy applied on the best model of each machine learning algorithm:

Table 20: Results of Trades profits with and without risk strategy

Stocks	Nb of Shares	SVM		SVR		DNN		RNN	
		Trades no Risk	Trades Risk	Trades no Risk	Trades Risk	Trades no Risk	Trades Risk	Trades no Risk	Trades Risk

AAPL	50	\$35,850	\$2,750	\$5,250	\$3,215	\$46,135	\$8,200	\$7,520	\$4,250
AMZN	6	\$35,580	\$582	\$21,894	\$180	\$2,322	\$90	\$10,950	\$73
GOOGL	8.5	\$847	\$2,644	\$1,139	\$2,797	\$2,669	\$2,797	\$205	\$2,412
FB	49	\$16,072	\$4,851	\$9,604	\$1,617	\$2,724	\$735	\$8,296	\$921

CHAPTER 6

CONCLUSION AND FUTURE WORK

An overview of the impacts and contributions of our proposed expert system:

In this thesis, we developed a stock price trend prediction system and automated trading system based on the risk strategy we defined. To build these models, we gathered data from two sources (i) historical stock market data from Reuters and (ii) news sentiment released to a related stock, also gathered from Reuters; this data was collected for 4 different stocks over 10 years. These data sources were used in recent studies and show the importance of integrating news with prices in order to make accurate predictions using the AI model. Technical features were calculated and used as input data to our model, in addition to 3 scenarios considered when adding sentiments to the calculated features. Our AI framework mainly incorporated DNN, RNN, SVR and SVM for prediction, which allowed us to select the best model for our feature's formalization and data sources.

From a research perspective, we consider our model innovative since we are trying to predict the price/trend of the closing price based on news data released today, and stock prices recorded on that day for the first n trading hours.

Our proposed prediction model is tested on APPL, AMZN, GOOGL and FB stock shares, for the data collected from (January 1, 2008 to December 31, 2017), resulting in 82.91% as the best accuracy.

In our research, we addressed the following theoretical questions:

- a) What is the value of adding news sentiments to stock prices? In contrast to other literature reviews, we selected a new timeframe in predicting the closing

price trend of the day and applied different new scenarios. Moreover, unlike other literatures where they discard multiple news released during the same interval, we used different scenarios to take this case into consideration, in addition to weekend news.

b) Do our data features produce good accuracy for short-term intervals or for longer intervals? After producing our features, we tried to predict the direction of the price movement for the next hour using the above different models, but this was not very successful with a 60% best accuracy level reached. On the other hand, when trying to predict the end-of-day price/movement, we reached 82.91% accuracy.

c) Which is the best new scenario to use in our AI model? Based on our study, we found that each stock will have a different window size and news scenario to be used for each model. This led us to conclude that news effects are different from one stock to another, and each stock has a different window size to predict the closing price. This is a testimonial to the idea that traders repeat, that each stock has a special movement that is repeated during specific frames.

d) Which model performs the best in terms of accuracy for stock price prediction? Adding news sentiments to featured data improves prediction accuracy. Our model analysis indicates that the closing price or trend with respect to yesterday's closing price can be predicted for different selected models better than prediction with a coin-flip. Moreover, our model shows a better performance for most of the models provided in the literature review.

1. Implementing Our Predictive Model

Our proposed model can be used in different ways. Firstly, our model can be used by traders who lack programming information; these traders can use our model either to predict the variation in price and for better analysis, or they can use our automated trading system without any monitoring – with the system opening and closing trades based on predictions. Secondly, our code can be easily deployed for short-term trading systems.

2. Future Work

After developing our model, a few enhancements could be done on our prediction model. One direction could be to add some famous technical indicators used in the stock market. Another direction could be to try different timeframes for grouping our data. And the final, and most important direction we could work on is to enhance the prediction of the exact price since for traders it is very different if the stock price is going to rise or fall by 20%, or by 1%.

In conclusion, this research has implemented a financial trading system that can predict stock price/trend. We have shown that by calculating different targets as output for the model, we can enhance prediction performance.

REFERENCES

- (1) Weng, B., Ahmed, M. A., & Megahed, F. M. (2017). Stock market one-day ahead movement prediction using disparate data sources. *Expert Systems with Applications*, 79, 153-163. doi:10.1016/j.eswa.2017.02.041
- (2) Li, X., Huang, X., Deng, X., & Zhu, S. (2014). Enhancing quantitative intra-day stock return prediction by integrating both market news and stock prices information. *Neurocomputing*, 142, 228-238. doi:10.1016/j.neucom.2014.04.043
- (3) Arévalo, A., Niño, J., Hernández, G., & Sandoval, J. (2016). High-Frequency Trading Strategy Based on Deep Neural Networks. *Intelligent Computing Methodologies Lecture Notes in Computer Science*, 424436. doi:10.1007/978-3-319-42297-8_40
- (4) Horne, J. C., & Parker, G. G. (1967). The Random-Walk Theory: An Empirical Test. *Financial Analysts Journal*, 23(6), 87-92. doi:10.2469/faj.v23.n6.87
- (5) Fama, E. F. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, 25(2), 383. doi:10.2307/2325486
- (6) Schumaker, R. P., & Chen, H. (2009). A quantitative stock prediction system based on financial news. *Information Processing & Management*, 45(5), 571-583. doi:10.1016/j.ipm.2009.05.001
- (7) Ding, X., Zhang, Y., Liu, T., & Duan, J. (2014). Using Structured Events to Predict Stock Price Movement: An Empirical Investigation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. doi:10.3115/v1/d14-1148
- (8) Schumaker, R. P., & Chen, H. (2009). A quantitative stock prediction system based on financial news. *Information Processing & Management*, 45(5), 571-583. doi:10.1016/j.ipm.2009.05.001
- (9) Bollen, J., & Mao, H. (2011). Twitter Mood as a Stock Market Predictor. *Computer*, 44(10), 91-94. doi:10.1109/mc.2011.323
- (10) Arora, A., et al.: *Deep Learning with H2O* (2015)
- (11) learning process 20. Zeiler, M.D.: ADADELTA: An Adaptive Learning Rate Method, 6 (2012)

(12) Singh, Aishwarya. "Predicting the Stock Market Using Machine Learning and Deep Learning." Analytics Vidhya, 26 July 2019, www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/.

(13) Mark L. Mitchell and J. Harold Mulherin The Journal of Finance Vol. 49, No. 3, Papers and Proceedings Fifty-Fourth Annual Meeting of the American Finance Association, Boston, Massachusetts, January 3-5, 1994 (Jul., 1994), pp. 923-950